Classical vs Quantum Random Oracles*

Takashi Yamakawa[†1] and Mark Zhandry[2,3]

[1]NTT Secure Platform Laboratories
[2]Princeton University
[3]NTT Research

March 5, 2021

Abstract

In this paper, we study relationship between security of cryptographic schemes in the random oracle model (ROM) and quantum random oracle model (QROM). First, we introduce a notion of a *proof of quantum access to a random oracle* (PoQRO), which is a protocol to prove the capability to quantumly access a random oracle to a classical verifier. We observe that a proof of quantumness recently proposed by Brakerski et al. (TQC '20) can be seen as a PoQRO. We also give a construction of a publicly verifiable PoQRO relative to a classical oracle. Based on them, we construct digital signature and public key encryption schemes that are secure in the ROM but insecure in the QROM. In particular, we obtain the first examples of natural cryptographic schemes that separate the ROM and QROM under a standard cryptographic assumption.

On the other hand, we give lifting theorems from security in the ROM to that in the QROM for certain types of cryptographic schemes and security notions. For example, our lifting theorems are applicable to Fiat-Shamir non-interactive arguments, Fiat-Shamir signatures, and Full-Domain-Hash signatures etc. We also discuss applications of our lifting theorems to quantum query complexity.

1 Introduction

The random oracle model (ROM) [BR93] is a widely used heuristic in cryptography where a hash function is modeled as a random function that is only accessible as an oracle. The ROM was used for constructing practical cryptographic schemes including digital signatures [FS87, PS96, BR96], chosen-ciphertext attack (CCA) secure public key encryption (PKE) [BR95, FOPS01, FO13], identity-based encryption (IBE) [GPV08], etc.

*This is a major update version of [YZ20] with many new results.
†This work was done while the author was visiting Princeton University.

In 2011, Boneh et al. [BDF+11] observed that the ROM may not be sufficient when considering post-quantum security, since a quantum adversary can quantumly evaluate hash functions on superpositions, while the ROM only gives a classically-accessible oracle to an adversary. Considering this observation, they proposed the quantum random oracle model (QROM), which gives an adversary quantum access to an oracle that computes a random function.

Boneh et al. observe that many proof techniques in the ROM cannot be directly translated into one in the QROM, *even if the other building blocks of the system are quantum-resistant*. Therefore, new proof techniques are needed in order to justify the post-quantum security of random oracle model systems. Fortunately, recent advances of proof techniques have clarified that most important constructions that are originally proven secure in the ROM are also secure in the QROM. These include OAEP [TU16], Fujisaki-Okamoto transform [TU16, JZC+18, Zha19], Fiat-Shamir transform [LZ19b, DFMS19, DFM20], Full-Domain Hash (FDH) signatures [Zha12], Gentry-Peikert-Vaikuntanathan (GPV) IBE [Zha12, KYY18], etc.

Given this situation, it is natural to ask if there may be a general theorem lifting *any* classical ROM proof into a proof in the QROM, provided that the other building blocks of the system remain quantum resistant. There are several known lifting theorems that ensure that certain types of security reductions in the ROM also work in the QROM [BDF+11, Son14, ZYF+19, KS20]. However, there is no known general lifting theorem that works regardless of form of security proofs in the ROM.

Such a general lifting theorem certainly seems like a challenging task. Nevertheless, demonstrating a separation — that is, a scheme using quantum-resistant building blocks that is secure in the ROM but insecure in the QROM — has also been elusive. Intuitively, the reason is that natural problems on random oracles (such as pre-image search, collision finding, etc) only have *polynomial* gaps between classical and quantum query complexity.

We are aware of two works that consider the task of finding a separation. First, Boneh et al. [BDF+11] gave an example of an identification protocol that is secure in the ROM but insecure in the QROM, but is specific to a certain non-standard timing model. Concretely, the protocol leverages the polynomial gap in collision finding to allow an attacker with quantum oracle access to break the system somewhat faster than any classical-access algorithm. The verifier then rejects if the prover cannot respond to its challenges fast enough, thereby blocking classical attacks while allowing the quantum attack to go through. This unfortunately requires a synchronous model where the verifier keeps track of the time between messages; such a model is non-standard.

Second, a recent work of Zhang et al. [ZYF+19] showed that quantum random oracle is *differentiable* from classical random oracle, which roughly means that it is impossible to simulate quantum queries to a random oracle using only classical queries to the same function. Their result rules out a natural approach one may take to give a lifting theorem, but it fails to actually give a scheme

separating classical from quantum access to a random oracle.[1]

In summary, there is no known classical cryptographic scheme (e.g., digital signatures or PKE) that can be proven secure in the ROM but insecure in the QROM. This leaves open the important question of whether or not a general lifting theorem for cryptographic schemes is possible.

1.1 Our Results

We give constructions of cryptographic schemes that separate the ROM and QROM, showing that a fully general lifting theorem is impossible. On the other hand, we also give lifting theorems from the ROM security to the QROM security for some constrained but still very general settings. Details are explained below:

1.1.1 Separation of ROM and QROM.

Proof of Quantum Access to a Random Oracle. For showing separations between the ROM and QROM, we first introduce a primitive which we call a *proof of quantum access to random oracle* (PoQRO). Roughly speaking, a PoQRO is a protocol where a quantum prover proves his ability to quantumly access to a random oracle to a classical verifier who is only given classical access to the random oracle. This is closely related to the notion of a proof of quantumness [BCM+18], but the difference is that a proof of quantumness only requires soundness against completely classical adversaries whereas a PoQRO requires soundness against *quantum* adversaries with classical access to a random oracle.

First, we observe that a proof of quantumness recently proposed by Brakerski et al. [BKVV20] is actually also a PoQRO. As a result, we obtain a PoQRO under the assumed quantum hardness of the learning with errors (LWE) problem [Reg09] (which we call the QLWE assumption in the following). The construction is non-interactive in the sense that after a verifier generates a pair of a public and secret keys and publishes the public key, a prover can generate a proof without any interaction. However, the proof is not publicly verifiable since the verification relies on the secret key.

We also study the possibility of publicly verifiable PoQRO. We give a construction of a publicly verifiable PoQRO relative to a classical oracle (which can be queried in superposition) using the technique developed in the recent work by Amos et al. [AGKZ20]. Similarly to [AGKZ20], we can heuristically instantiate the protocol in the standard model by using candidate constructions of post-quantum obfuscation [Agr19, AP20, BDGM20, WW20, GP20].

Separation of ROM and QROM. A PoQRO itself is already an example of cryptographic task that can be done in the QROM but cannot be done in the ROM. By embedding a PoQRO into digital signatures and PKE, we obtain the following results:

[1] Subsequent to the posting of the initial version of this work online [YZ20], Zhang et al. [ZYF+19] updated their paper to add a construction of a cryptographic scheme that separates the ROM and the QROM. See Sec. 1.3 for details.

- A digital signature scheme that is EUF-CMA secure in the ROM but completely broken by 1 signing query in the QROM, and
- A PKE scheme that is IND-CCA secure in the ROM but completely broken by 1 decryption query in the QROM.

Both these results rely on the QLWE assumption.

Moreover, by embedding a publicly verifiable PoQRO into them, we can show the existence of a classical oracle relative to which there exist the following schemes:

- A digital signature scheme that is EUF-CMA secure in the ROM but not even EUF-NMA secure[2] in the QROM, and
- A PKE scheme that is IND-CCA secure in the ROM but not even IND-CPA secure in the QROM.

These results can be understood as an evidence that a generic lifting theorem is unlikely to exist even for the weak security notions of EUF-NMA security of digital signatures and IND-CPA security of PKE. Specifically, the above results imply that there do not exist a relativizing lifting theorem for them that works relative to any classical oracle.

1.1.2 Lifting Theorems

We now turn to our positive results, giving lifting theorems for certain class of schemes and security notions.

Lifting Theorem for Search-Type Games. First, we give a lifting theorem for what we call *search-type games*. A search-type game is specified by a classical challenger that interacts with an adversary and finally outputs $\top$ indicating acceptance or $\bot$ indicating rejection. We say that the adversary wins if the verifer outputs $\top$. We say that the game is hard in the ROM (resp. QROM) if no efficient quantum adversary with classical (resp. quantum) access to the random oracle can win the game with non-negligible probability. For example, the soundness of PoQROs is captured by the hardness of a search-type game in the ROM (but not QROM!), and the EUF-CMA/NMA security of digital signatures in the ROM (resp. QROM) is captured by the hardness of a search-type game in the ROM (resp. QROM). We prove the following theorem:

Theorem 1.1 (Lifting Theorem for Search-Type Game, Informal)**.** *For any search-type game where a challenger makes* constant *number of queries to the random oracle, if the game is hard in the ROM, then that is also hard in the QROM.*

As immediate corollaries of the theorem, we obtain lifting theorems for the following:

[2]The EUF-NMA security is an unforgeability against adversaries that do not make any signing query. See Definition 2.3.

- EUF-NMA security of digital signatures whose key generation and verification algorithms make $O(1)$ random oracle queries, and
- Soundness of (non-)interactive arguments whose (setup algorithm and) verifier make at most $O(1)$ random oracle queries.

The latter lifting theorem is applicable to those obtained by the Fiat-Shamir transform to constant round interactive arguments. Though it is already proven that such arguments are sound in the QROM [LZ19b, DFMS19, DFM20], we believe that the above general corollary would be still useful for the design of non-interactive arguments in the QROM in the future without repeating a similar analyses to those works.

Theorem 1.1 also immediately implies the impossibility of PoQRO where the verifier makes $O(1)$ random oracle queries. We note that in our PoQRO protocols, the number of queries made by the verification algorithm is $\omega(\log \lambda)$. We leave it as an interesting open problem to study the (im)possibility of PoQRO with $O(\log \lambda)$-query verification.

Though the applicability of Theorem 1.1 is somewhat limited, to the best of our knowledge, this is the first general lifting theorem from ROM security to QROM security that does not make any assumptions about the ROM security reduction.

Lifting Theorem for EUF-CMA Security of Digital Signatures. Unfortunately, Theorem 1.1 does not give a lifting theorem for the EUF-CMA security of digital signatures (except for a non-interesting case where the signing algorithm does not make random oracle query). On the other hand, we give a lifting theorem for the EUF-CMA security for digital signature shcmes that satisfy additional properties.

Theorem 1.2 (Lifting Theorem for Digital Signatures, Informal)**.** *Suppose that a digital signature scheme satisfies the following:*

1. *EUF-NMA secure in the ROM,*
2. *The key generation algorithm does not make random oracle queries and the verification algorithm makes $O(1)$ random oracle queries,*
3. *Random oracle queries made by the signing and verification algorithms reveal the corresponding message, and*
4. *Signatures are simulatable without the signing key if one is allowed to non-adaptively program the random oracle.*

Then the scheme is EUF-CMA secure in the QROM.

This theorem is applicable to the FDH signatures and Fiat-Shamir signatures. To the best of our knowledge, this is the first lifting theorem that is simultaneously applicable to both of them.

Application to Quantum Query Complexity. Based on a slight variant of a quantitative version of Theorem 1.1, we obtain a general theorem about query complexity. We consider a class of oracle problems, where the adversary's goal is to find distinct inputs to H such that the corresponding outputs satisfy some relation. Our theorem can be seen as upper bounding the success probability of a q-query adversary in terms of the probability of an adversary that makes no queries at all. Slightly more formally:

Theorem 1.3 (Informal). *Let $H : \mathcal{X} \to \mathcal{Y}$ be a random oracle. For any relation $R \subseteq \mathcal{Y}^k$, the probability that a q-quantum-query adversary finds pairwise distinct $x_1, ..., x_k$ such that $(H(x_1), ..., H(x_k)) \in R$ is at most*

$$(2q+1)^{2k} \Pr[\exists \pi \text{ s.t. } (y_{\pi(1)}, ..., y_{\pi(k)}) \in R : (y_1, ..., y_k) \xleftarrow{\$} \mathcal{Y}^k] \tag{1}$$

where π is a permutation over $\{1, ..., k\}$.

The probability in Equation 1 is typically be very easy to analyze. Theorem 1.3 therefore yields very simple non-trivial query lower bounds for various problems including (multi-)preimage search and (multi- or generalized) collision finding. Though these bounds are already known and/or non-tight, an advantage of our proofs is its extreme simplicity once we have Theorem 1.3 in hand.

1.2 Technical Overview

PoQRO from LWE. We first observe that a proof of quantumness in [BKVV20] is also a PoQRO. Though the construction and security proof are essentially the same as theirs, we briefly review them for the reader's convenience. The protocol is based on a noisy trapdoor claw-free permutation constructed from the QLWE assumption [BCM+18, BKVV20]. In this overview, we assume that there is a clean trapdoor claw-free permutation for simplicity. A claw-free permutation is a function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ such that (1) $f(0,\cdot)$ and $f(1,\cdot)$ are injective, (2) it is difficult for an efficient quantum adversary given f to find a claw (x_0, x_1) such that $f(0,x_0) = f(1,x_1)$, but (3) there is a trapdoor that enables one to efficiently find both pre-images for any target value. Let H be a random oracle from $\{0,1\}^n$ to $\{0,1\}$. In the PoQRO, the verifier first generates f along with its trapdoor and only sends f to the prover as a public key. Then the prover generates a state $\frac{1}{2}(|0\rangle |x_0\rangle + |1\rangle |x_1\rangle)$ along with $y = f(0,x_0) = f(1,x_1)$ by using the technique of [BCM+18]. Then it applies the random oracle H into the phase to get $\frac{1}{2}((-1)^{H(x_0)} |0\rangle |x_0\rangle + (-1)^{H(x_1)} |1\rangle |x_1\rangle)$, applies the Hadamard transform, measures both registers to obtain (m,d), and sends (y,m,d) as a proof to the verifier. The verifier computes x_0 and x_1 from y by using the trapdoor and accepts if $m = d^T \cdot (x_0 \oplus x_1) \oplus H(x_0) \oplus H(x_1)$ holds. As shown in [BKVV20], the equation is satisfied if the prover honestly run the protocol. On the other hand, a cheating prover with classical access to H can pass the test with probability almost $1/2$ since the only way to obtain an information of $H(x_0) \oplus H(x_1)$ is to query both x_0 and x_1; this happens with

a negligible probability due to the claw-free property. This construction only gives a constant gap between completeness and soundness, so we amplify it to super-polynomial by $\omega(\log \lambda)$ parallel repetitions.

Publicly Verifiable PoQRO. We construct a publicly verifiable PoQRO based on a variant of an equivocal collision-resistant hash (ECRH) [AGKZ20]. An ECRH $f : \mathcal{X} \to \mathcal{Y}$ is a collision-resistant hash function with a special property called equivocality. The equivocality enables one to generate a pair of a classical string $y \in \mathcal{Y}$ and a quantum state $|\mathsf{sk}\rangle$ that can be used to find x such that $f(x) = y$ and $p(x) = b$ where $p : \mathcal{X} \to \{0, 1\}$ is a pre-determined predicate and b is a bit chosen after $(y, |\mathsf{sk}\rangle)$ is generated. Amos et al. [AGKZ20] constructed an ECRH for a predicate p that returns the first bit of its input relative to a classical oracle. Here, we observe that their construction can be extended to support *any* predicate p. Specifically, we can define p as a predicate defined by a random oracle $H : \mathcal{X} \to \{0, 1\}$. Based on such an ECRH, we can construct a 4-round publicly verifiable PoQRO as follows:

1. The verifier generates an ECRH f and sends f to the prover.
2. The prover generates y along with the corresponding $|\mathsf{sk}\rangle$ and sends y to the verifier
3. The verifier randomly chooses $b \xleftarrow{\$} \{0, 1\}$ and sends b to the prover.
4. The prover finds x such that $f(x) = y$ and $H(x) = b$ by using $|sk\rangle$ and sends x to the verifier.
5. The verifier accepts if and only if $f(x) = y$ and $H(x) = b$.

By the functionality of ECRH, the verifier accepts with overwhelming probability if a prover with quantum access to H runs honestly. On the other hand, if a cheating prover is given only classical access to H, then the verifier will accept with probability almost $1/2$. To see this, consider the first query the prover makes to H on an x^* such that $f(x^*) = y$. If the prover ultimately sends an $x \neq x^*$ to the verifier that causes the verifier to accept, x and x^* will be a collision for f, contradicting the collision-resistance of f. On the other hand, if $x = x^*$, then $H(x) = H(x^*)$ has only a $1/2$ chance of being equal to b, regardless of whether the query on x^* happened before or after the prover learned b. The result is that, no matter what the prover does, the verifier rejects with probability essentially at least $1/2$.

This protocol only achieves a constant gap between completeness and soundness, but it can be amplified to super-polynomial by $\omega(\log \lambda)$ parallel repetitions. Moreover since the verifier's message in the third round is just a public coin, we can apply the Fiat-Shamir transform to the above protocol to make the protocol non-interactive considering the generation of f as a setup.

Separations for Digital Signatures and Public Key Encryption. Given a PoQRO, it is easy to construct digital signature and PKE schemes that are

secure in the ROM but insecure in the QROM: Suppose that we have a EUF-CMA secure digital signature scheme in the ROM, consider a modified scheme in which the signing algorithm returns a secret key of the scheme if the queried message is a valid proof of the PoQRO. Clearly, this scheme is insecure in the QROM and completely broken by 1 signing query. On the other hand, security in the ROM is preserved since an adversary in the ROM cannot find a valid proof of the PoQRO. A separation for IND-CCA security of PKE can be obtained by embedding verification of PoQRO in a decryption algorithm in a similar manner.

Moreover, if the PoQRO is publicly verifiable, then we can embed the verification of the PoQRO into verification and encryption algorithms of digital signature and PKE schemes, respectively. As a result, we obtain separations even for EUF-NMA secure digital signatures and IND-CPA secure PKE schemes, assuming an equivocal collision-resistant hash function.

Lifting Theorem for Search-Type Games. Next, we give a brief overview of proofs of our lifting theorems. A starting point of our lifting theorem is the following *classical* lemma:

Lemma 1.4. *(Informal) For any search-type cryptographic game in which a challenger makes at most k classical random oracle queries, if there exists an efficient adversary $\mathcal{A}$ that makes at most q classical random oracle queries with winning probability ϵ, then there exists an efficient $\mathcal{B}$ that makes at most k classical random oracle queries with winning probability at least $\epsilon/(q+1)^k$.*

This lemma can be proven by considering $\mathcal{B}$ described as follows:

1. Let H be the "real" random oracle that is given to $\mathcal{B}$.
2. For each $j = 1, ..., k$, $\mathcal{B}$ randomly picks $i_j \xleftarrow{\$} [q+1]$. Intuitively, this is a guess of $\mathcal{A}$'s first query that is equal to the challenger's j-th query where $i_j = q+1$ is understood as a guess that "$\mathcal{A}$ does not make such a query".
3. $\mathcal{B}$ chooses a fresh "fake" random oracle H' by itself.[3]
4. $\mathcal{B}$ runs $\mathcal{A}$ by giving $\mathcal{A}$ a stateful oracle $\mathcal{O}$ simulated as follows: $\mathcal{B}$ initializes $\mathcal{O}$ to H'. Whenever $\mathcal{A}$ makes its i-th query x_i, $\mathcal{B}$ simulates the oracle $\mathcal{O}$ in one of the following ways:
   (a) If $i = i_j$ for some $j \in [k]$, then $\mathcal{B}$ queries x_i to the real random oracle H to obtain $H(x_i)$, returns $H(x_i)$, and reprograms $\mathcal{O}$ to output $H(x_i)$ on input x_i.
   (b) Otherwise, $\mathcal{B}$ just returns $\mathcal{O}(x_i)$.

   Whenever $\mathcal{A}$ sends some message to the challenger, $\mathcal{B}$ just forwards it to the external challenger, and whenever the challenger returns some message, $\mathcal{B}$ forwards it to $\mathcal{A}$.

[3] More precisely, it simulates a fresh random oracle H' on the fly so that this can be done efficiently. Alternatively, it can choose H' from a family of q-wise independent functions.

Clearly, $\mathcal{B}$ makes at most k classical random oracle queries and is as efficient as $\mathcal{A}$. We can see that $\mathcal{B}$ perfectly simulates the game for $\mathcal{A}$ if the guess is correct (e.g., $\mathcal{A}$'s i_j-th query is its first query that is equal to the challenger's j-th query), which happens with probability $1/(q+1)^k$. Moreover, since the events that the guess is correct and the event that $\mathcal{A}$ wins are independent, we can conclude that $\mathcal{B}$'s winning probability is at least $1/(q+1)^k$ times $\mathcal{A}$'s winning probability.

Our idea is to apply a similar proof to $\mathcal{A}$ that may make quantum queries, with the goal of $\mathcal{B}$ still only needing classical queries. Then, an obvious problem is that $\mathcal{B}$ cannot forwards $\mathcal{A}$'s query in Step 4a since $\mathcal{A}$'s query may be quantum whereas $\mathcal{B}$ only has classical access to the real random oracle H. Here, our solution is to just let $\mathcal{B}$ measure $\mathcal{A}$'s query, query the measurement outcome to the real random oracle H, and then reprogram $\mathcal{O}$ according to this value. Of course, such a measurement can be noticed by $\mathcal{A}$ by a noticeable advantage. Nonetheless, we can rely on the techniques developed for Fiat-Shamir transform in the QROM [DFMS19, DFM20] to prove that this decreases the winning probability only by the factor of $(2q+1)^{2k}$. Therefore, as long as $k = O(1)$, the reduction works with a polynomial loss.

Application to Digital Signatures. Our lifting theorem for search-type games (Theorem 1.1) immediately implies a lifting theorem for EUF-NMA security for digital signature schemes where key generation and verification algorithms make constant number of random oracle queries. On the other hand, Kiltz et al. [KLS18] showed that the EUF-NMA security in the QROM implies EUF-CMA security in the QROM for Fiat-Shamir signatures. We generalize this result to a broader class of digital signature schemes that satisfy conditions given in Theorem 1.2. Roughly speaking, this can be proven based on the observation that if signatures are simulatable without the signing key by programming the random oracle, then the signing oracle is useless and thus the EUF-NMA and EUF-CMA security are equivalent. By combining this with Theorem 1.1, we obtain Theorem 1.2.

Application to Quantum Query Complexity. As one can see from the overview of the proof of Theorem 1.1, the security loss of the reduction from QROM adversary to ROM adversary is $(2q+1)^{2k}$. By applying a (slight variant of) this quantitative version of Theorem 1.1 to a search-type game to find a pairwise distinct $(x_1, ..., x_k)$ such that $(H(x_1), ..., H(x_k)) \in R$, we obtain Theorem 1.3.

1.3 Related Works

P versus BQP relative to a random oracle. As a related question to the topic of this paper, Fortnow and Rogers [FR99] asked if we can separate complexity classes **P** and **BQP** relative to a random oracle. Though Aaronson and Ambainis [AA14] gave an evidence that it is difficult to separate (an average case version of) **P** and **BQP** relative to a random oracle under a certain conjec-

ture, an unconditional proof is still open. We note that our separations between ROM and QROM do not give any implication to the problem since we rely on computational assumptions and consider an interactive protocol, which cannot be captured as a decision problem.

Separation of ROM and QROM for Sampling. Aaronson [Aar10] showed that there is a sampling problem (called Fourier Sampling) that can be solved by 1 quantum query to a random oracle but requires exponential number of classical queries. We note that this does not give a separation of the ROM and QROM in a cryptographic setting since a classical verifier cannot efficiently check that the sample is taken according to the correct distribution.

Known Lifting Theorems. Though several works [BDF+11, Son14, ZYF+19, KS20] give lifting theorems from ROM security to QROM security, they assume certain conditions for security proofs in the ROM. On the other hand, our lifting theorem for search-type games only requires syntactic conditions of schemes and their security notions, and do not assume anything about security proofs in the ROM. Our lifting theorem for digital signatures requires slightly more involved conditions, but we believe that it is much easier to check them than to check that a security proof in the ROM relies on a certain type of reductions.

Quantum Query Complexity. Beals et al. [BBC+01] showed that quantum query complexity is polynomially related to classical query complexity for any total functions. Though this may seem closely related to our result on query complexity, there are two significant differences. First, they consider a problem to output a 1-bit predicate considering the oracle as an input, whereas we consider a problem to find k inputs whose oracle values are in a certain relation. Second, they consider the worst case complexity whereas we consider the average case complexity. Due to the above two differences, these two results are incomparable.

Zhandry [Zha19, Theorem 3] also gave a general theorem that gives average case quantum query lower bounds relative to a random oracle. Their theorem gives tighter lower bounds than ours for some problems (e.g., collision finding). On the other hand, we believe that ours is easier to apply and also more general than theirs. For example, their theorem does not (at least directly) give meaningful lower bounds for the generalized collision finding problems.

Concurrent Work. Subsequent to the posting of the initial version of this work online [YZ20], Zhang et al. [ZYF+19] updated their paper to add a construction of (an interactive version of) PoQRO based on the QLWE assumption. Their construction is based on an ad hoc modification of Mahadev's classical verification of quantum computation protocol [Mah18], and completely different from ours.

2 Preliminaries

Notations. We use λ to mean the security parameter throughout the paper. For a set X, $|X|$ is the cardinality of X. We denote by $x \xleftarrow{\$} X$ to mean that we take x uniformly from X. For sets $\mathcal{X}$ and $\mathcal{Y}$, $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ denotes the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. For a positive integer n, $[n]$ means a set $\{1, ..., n\}$. We say that a quantum (resp. classical) algorithm is efficient if that runs in quantum (resp. classical) polynomial time. For a quantum or randomized classical algorithm $\mathcal{A}$, we denote $y \xleftarrow{\$} \mathcal{A}(x)$ to mean that $\mathcal{A}$ outputs y on input x, and denote $y \in \mathcal{A}(x)$ to mean that y is in the support of $\mathcal{A}(x)$.

Oracles. In this paper, we consider the following three types of oracles: quantum oracle, quantumly-accessible classical oracle, and classically-accessible classical oracle.

A *quantum oracle* is an oracle that applies a unitary U on a query register. A *quantumly-accessible classical oracle* is a special case of a quantum oracle where U computes a classical function, i.e., there exists a classical function f such that we have $U|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$ for any x and y in the domain and range of f. By a standard technique, when f is a single-bit output function, we can implement an oracle that applies a unitary U' such that $U'|x\rangle = (-1)^{f(x)}|x\rangle$ for any x by a single call to an oracle that applies U as above. We call an oracle that applies U' a phase oracle of f. A *classically-accessible classical oracle* works similarly to a quantumly-accessible classical oracle except that it measures the first register (the register to store x) in standard basis in each query. When we just say that an oracle is a classical oracle, then that is quantumly-accessible for any quantum algorithm and classically-accessible for any classical algorithm. For an oracle-aided quantum algorithm $\mathcal{A}$ and a classical function f, we often denote by $\mathcal{A}^{|f\rangle}$ (resp. $\mathcal{A}^{f}$) to mean that $\mathcal{A}$ is given a quantumly-accessible (resp. classically-accessible) classical oracle that computes f.

Classical/Quantum Random Oracle Model. In the (classical) random oracle model (ROM) [BR93], a random function H (of a certain domain and range) is chosen at the beginning, and every party (including honest algorithms of a protocol whose security is analyzed and an adversary) can classically access H. In other words, they are given a classically-accessible classical oracle that computes H. The quantum random oracle model (QROM) [BDF+11] is defined similarly except that the access to H can be quantum. In other words, a quantumly-accessible classical oracle that computes H is available for the adversary.[4] We stress that the classical ROM can be considered even when we consider security against quantum adversaries. We say that an algorithm in the QROM (resp. ROM) is q-quantum-query (resp. q-classical-query) if it makes at most q queries to its oracle.

By the following lemma, we can efficiently simulate a quantum random oracle

[4] Since we consider the post-quantum setting where honest algorithms are classical, the only party who may quantumly access H is the adversary.

to a q-quantum-query algorithms by using $2q$-wise independent hash function.[5]

Lemma 2.1 ([Zha12]). *For any sets $\mathcal{X}$ and $\mathcal{Y}$ of classical strings and q-quantum-query algorithm $\mathcal{A}$, we have*

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

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

Oracle Indistinguishability Lemma. We will use the following lemma.

Lemma 2.2. *(Special case of [BZ13, Lemma 2.5]) Let $\mathcal{X}$ and $\mathcal{Y}$ be sets, let D and D' be statistically indistinguishable distributions on $\mathcal{Y}$. Let $\mathcal{H}_D$ (resp. $\mathcal{H}_{D'}$) be a distribution of $H : \mathcal{X} \to \mathcal{Y}$ that, for each $x \in \mathcal{X}$, sets $H(x)$ to be a value drawn from D (resp. D') independently. Then for any $\mathsf{poly}(\lambda)$-quantum-query algorithm $\mathcal{A}$, we have*

$$\left|\Pr[\mathcal{A}^{|H\rangle} = 1 : H \xleftarrow{\$} \mathcal{H}_D] - \Pr[\mathcal{A}^{|H\rangle} = 1 : H \xleftarrow{\$} \mathcal{H}_{D'}]\right| = \mathsf{negl}(\lambda).$$

Learning with Errors. Roughly speaking, a learning with errors (LWE) [Reg09] problem is a problem to solve a system of noisy linear equations. Regev [Reg09] gave a quantum reduction from hardness of LWE to hardness of worst-case lattice problems, and it has been conjectured that the LWE problem is hard to solve in quantum polynomial time. We call the assumption that no quantum polynomial time algorithm can solve the LWE problem QLWE assumption. We omit a detailed definition and a concrete parameter choice for the LWE problem since we use the QLWE assumption only as a building block for constructing general primitives such as noisy trapdoor claw-free functions [BCM+18, BKVV20], PKE [Reg09, PW08], and digital signatures [GPV08]. We refer to these works for concrete parameter choices.

Cryptographic Primitives. Here, we define standard cryptographic primitives and its security.

Definition 2.3 (Digital Signatures.). *A digital signature scheme consists of classical algorithms* $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$*:*

$\mathsf{Sig.KeyGen}(1^\lambda)$**:** *This algorithm takes the security parameter 1^λ as input and outputs a verification key* vk *and a signing key* sigk*.*

$\mathsf{Sig.Sign}(\mathsf{sigk}, m)$**:** *This algorithm takes a signing key* sigk *and a message m as input and outputs a signature σ.*

$\mathsf{Sig.Verify}(\mathsf{vk}, m, \sigma)$**:** *This algorithm takes a verification key* vk*, a message m, and a signature σ as input, and outputs $\top$ indicating acceptance or $\bot$ indicating rejection.*

[5] Though Zhandry [Zha19] gives another method to simulate a quantum random oracle without upper bounding the number of queries, we use a simulation by $2q$-wise independent hash functions for simplicity.

As correctness, we require that for any m, we have

$$\Pr[\mathsf{Sig.Verify}(\mathsf{vk}, x, \sigma) = \top : (\mathsf{vk}, \mathsf{sigk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda), \sigma \xleftarrow{\$} \mathsf{Sig.Sign}(\mathsf{sigk}, m)] = 1.$$

We say that a digital signature scheme is n-EUF-CMA secure against quantum adversaries if for any efficient quantum adversary $\mathcal{A}$ that makes at most n queries to the signing oracle, we have

$$\Pr\left[\begin{array}{l|ll} \mathsf{Sig.Verify}(\mathsf{vk}, m^*, \sigma^*) = \top & : & (\mathsf{vk}, \mathsf{sigk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda), \\ \wedge\ \mathcal{A} \textit{ never queried } m^* & & (m^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathsf{Sig.Sign}(\mathsf{sigk}, \cdot)}(\mathsf{vk}) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Sig.Sign}(\mathsf{sigk}, \cdot)$ denotes a classically-accessible classical oracle that computes $\mathsf{Sig.Sign}(\mathsf{sigk}, \cdot)$.

We say that a digital signature scheme is EUF-CMA secure if it is n-EUF-CMA-secure for all $n = \mathsf{poly}(\lambda)$. We say that a digital signature scheme is EUF-NMA secure if it is 0-EUF-CMA-secure.

Definition 2.4 (Public Key Encryption)**.** *A public key encryption (PKE) scheme consists of classical polynomial time algorithms* $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$*:*

$\mathsf{PKE.KeyGen}(1^\lambda)$**:** *This algorithm takes the security parameter 1^λ as input and outputs an encryption key* ek *and a decryption key* dk*.*

$\mathsf{PKE.Enc}(\mathsf{ek}, m)$**:** *This algorithm takes an encryption key* ek *and a message m as input and outputs a ciphertext* ct*.*

$\mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{ct})$**:** *This algorithm takes a decryption key* dk *and a ciphertext* ct *as input and outputs a message m or $\bot$.*

As correctness, we require that for any m, we have

$$\Pr[\mathsf{PKE.Dec}(\mathsf{dk}, \mathsf{ct}) = m : (\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda), \mathsf{ct} \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}, m)] = 1.$$

We say that a PKE scheme is n-IND-CCA secure against quantum adversaries if for any quantum polynomial-time adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that makes at most n classical queries to the decryption oracle, we have

$$\left|\Pr\left[\begin{array}{l|ll} \mathcal{A}_2^{\mathsf{PKE.Dec}(\mathsf{dk}, \cdot)}(|\mathsf{st}\rangle, \mathsf{ct}^*) = b & : & (\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}(1^\lambda), \\ \wedge\ \mathcal{A}_2 \textit{ never queried } \mathsf{ct}^* & & (m_0, m_1, |\mathsf{st}\rangle) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{PKE.Dec}(\mathsf{dk}, \cdot)}, \\ & & b \xleftarrow{\$} \{0,1\}, \\ & & \mathsf{ct}^* \xleftarrow{\$} \mathsf{PKE.Enc}(\mathsf{ek}, m_b) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{PKE.Dec}(\mathsf{dk}, \cdot)$ denotes a classically-accessible classical oracle that computes $\mathsf{PKE.Dec}(\mathsf{dk}, \cdot)$.

We say that a PKE scheme is IND-CCA secure if it is n-IND-CCA secure for all $n = \mathsf{poly}(\lambda)$. We say that a PKE scheme is IND-CPA secure if it is 0-IND-CCA secure.

3 Separation between ROM and QROM

In this section, we show examples of cryptographic schemes that are secure in the ROM but insecure in the QROM.

3.1 Proof of Quantum Access to Random Oracle

First, we introduce a notion of proofs of quantum access to a random oracle (PoQRO).

Definition 3.1. *A (non-interactive) proof of quantum access to a random oracle (PoQRO) consists of algorithms* $(\mathsf{PoQRO.Setup}, \mathsf{PoQRO.Prove}, \mathsf{PoQRO.Verify})$.

$\mathsf{PoQRO.Setup}(1^\lambda)$**:** *This is a classical algorithm that takes the security parameter* 1^λ *as input and outputs a public key* pk *and a secret key* sk.

$\mathsf{PoQRO.Prove}^{|H\rangle}(\mathsf{pk})$**:** *This is a quantum oracle-aided algorithm that takes a public key* pk *as input and given a quantum access to a random oracle* H*, and outputs a proof* π.

$\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, \pi)$**:** *This is a classical algorithm that takes a secret key* sk *and a proof* π *as input and given a classical access to a random oracle* H*, and outputs* $\top$ *indicating acceptance or* $\bot$ *indicating rejection.*

We require PoQRO to satisfy the following properties.

Correctness. *We have*

$$\Pr\left[\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, \pi) = \bot : \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PoQRO.Setup}(1^\lambda), \\ \pi \xleftarrow{\$} \mathsf{PoQRO.Prove}^{|H\rangle}(\mathsf{pk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Soundness. *For any quantum polynomial-time adversary* $\mathcal{A}$ *that is given a classical oracle access to* H*, we have*

$$\Pr\left[\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, \pi) = \top : \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PoQRO.Setup}(1^\lambda), \\ \pi \xleftarrow{\$} \mathcal{A}^{H}(\mathsf{pk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 3.2 (Public Verifiability)**.** *We say that PoQRO is publicly verifiable if we have* $\mathsf{pk} = \mathsf{sk}$ *for any* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{PoQRO.Setup}$*. When we consider a publicly verifiable PoQRO, we omit* sk *from the output of the setup algorithm and gives* pk *instead of* sk *to the verification algorithm for notational simplicity.*

3.1.1 PoQRO from QLWE

We observe that proofs of quantumness recently proposed by Brakerski et al. [BKVV20] can also be seen as PoQRO. Specifically, by just replacing "classical prover" with "quantum prover with classical access to the random oracle", their security proof directly works as a security proof of PoQRO.

Theorem 3.3 (a variant of [BKVV20])**.** *If the QLWE assumption holds, then there exists a PoQRO.*

Proof. (sketch) Though the construction and security proof are essentially the same as those in [BKVV20], we give a proof sketch for the reader's convenience. As shown in previous works [BCM+18, BKVV20] there exists a quantumly secure family of noisy trapdoor claw-free functions assuming the QLWE assumption. In this proof sketch, we assume that there exists a quantumly-secure family of (non-noisy) trapdoor claw-free functions for simplicity. We note that the proof can be easily extended to the construction from a noisy one as in [BKVV20].

A quantumly secure family of trapdoor claw-free functions enables one to sample a function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ along with a trapdoor such that

1. $f(0,\cdot)$ and $f(1,\cdot)$ are injective,
2. $f(0,\cdot)$ and $f(1,\cdot)$ are efficiently invertible by using a trapdoor, and
3. it is hard for an efficient quantum adversary that is not given a trapdoor to find x_0 and x_1 such that $f(0,x_0) = f(1,x_1)$.

Let $H : \{0,1\}^n \to \{0,1\}$ be a random oracle. First, we describe a PoQRO with soundness error $1/2$.

$\mathsf{PoQRO.Setup}(1^\lambda)$**:** This algorithm generates a trapdoor claw-free function $f : \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ along with a trapdoor td, and outputs $\mathsf{pk} := f$ and $\mathsf{sk} := \mathsf{td}$.

$\mathsf{PoQRO.Prove}^{|H\rangle}(\mathsf{pk} = f)$**:** This algorithm generates a superposition

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{2^{n/2}} \sum_{x\in\{0,1\}^n} |x\rangle\,,$$

computes f into another register to obtain

$$\frac{1}{2^{(n+1)/2}} \left(|0\rangle \sum_{x\in\{0,1\}^n} |x\rangle\,|f(0,x)\rangle + |1\rangle \sum_{x\in\{0,1\}^n} |x\rangle\,|f(1,x)\rangle \right),$$

measures the third register to obtain $y \in \{0,1\}^n$ along with a collapsed state

$$\frac{1}{\sqrt{2}}(|0\rangle\,|x_0\rangle + |1\rangle\,|x_1\rangle)$$

where $f(0,x_0) = f(1,x_1) = y$, applies the phase oracle of H on the second register to obtain

$$\frac{1}{\sqrt{2}}((-1)^{H(x_0)}\,|0\rangle\,|x_0\rangle + (-1)^{H(x_1)}\,|1\rangle\,|x_1\rangle),$$

applies the Hadamard transform on both registers to obtain

$$\frac{1}{2^{(n+2)/2}}\sum_{((m,d)\in\{0,1\}\times\{0,1\}^n)}((-1)^{H(x_0)\oplus d^Tx_0}+(-1)^{H(x_1)\oplus m\oplus d^Tx_1})\,|m\rangle\,|d\rangle$$
$$=\frac{1}{2^{n/2}}\sum_{(m,d):m=d^T\cdot(x_0\oplus x_1)\oplus H(x_0)\oplus H(x_1)}(-1)^{H(x_0)\oplus d^Tx_0}\,|m\rangle\,|d\rangle\,,$$

and measures the both registers in standard basis to obtain (m,d). Then it outputs $\pi := (y,m,d)$.

$\mathsf{PoQRO.Verify}^H(\mathsf{sk}=\mathsf{td},\pi=(y,m,d))$**:** This algorithm computes x_0 and x_1 such that $f(0,x_0)=f(1,x_1)=y$ by using a trapdoor td and outputs $\top$ if

$$m = d^T\cdot(x_0\oplus x_1)\oplus H(x_0)\oplus H(x_1)$$

holds and $\bot$ otherwise.

The correctness clearly follows from the above description. For proving soundness, we consider an efficient quantum adversary $\mathcal{A}^H$ that is given classical access to H. First, it is easy to see that $\mathcal{A}$ can win with probability $1/2$ if it does not query both x_0 and x_1 to H. Moreover, if $\mathcal{A}$ queries both x_0 and x_1 to H, then we can break the security of the trapdoor claw-free function f by finding a solution from $\mathcal{A}$'s queries.[6] Therefore, such an event happens with negligible probability, and thus $\mathcal{A}$'s winning probability is at most $1/2+\mathsf{negl}(\lambda)$. Finally, by a $\omega(\log(\lambda))$ parallel repetition, we can exponentially reduce the soundness error to obtain a PoQRO with negligible soundness error.[7] □

3.1.2 Publicly Verifiable PoQRO relative to Classical Oracle

Next, we give a construction of a publicly verifiable PoQRO relative to a classical oracle based on a variant of equivocal collision-resistant hash functions recently introduced in [AGKZ20].

Theorem 3.4. *There exists a publicly verifiable PoQRO relative to a quantumly-accessible classical oracle that is independent of the random oracle.*

Remark 1. *One may think that we can upgrade any PoQRO to publicly verifiable one by just relativizing to a classical oracle in which* sk *is hardwired that runs the verification algorithm. However, in such a construction, the classical oracle depends on the random oracle, which we believe is not desirable. Especially, such a construction cannot be instantiated in the standard model even assuming an ideal obfuscation since we do not know how to obfuscate a circuit*

[6] Such an extraction of $\mathcal{A}$'s queries can be done since we assume that $\mathcal{A}$ only *classically* access to the random oracle.

[7] It may not be immediately clear that a parallel repetition exponentially decreases the soundness error. For a formal proof, we can use the same game hops as in [BKVV20]. We note that they give a proof for the case of λ parallel repetitions, but $\omega(\log(\lambda))$ repetitions suffice.

with random oracle gates. On the other hand, we consider a construction relative to a classical oracle that does not depend on the random oracle, which enables us to heuristically instantiate the construction in the standard model by using an obfuscation.

For proving Theorem 3.4, we introduce a slightly stronger variant of equivocal collision-resistant hash functions [AGKZ20].

Definition 3.5 (Equivocal Collision-Resistant Hash Functions for General Predicates)**.** *An equivocal collision-resistant hash function (ECRH) family for general predicates with a domain $\mathcal{X}$ and a range $\mathcal{Y}$ is a tuple* $(\mathsf{ECRH.Setup}, \mathsf{ECRH.Gen}, \mathsf{ECRH.Eval}, \mathsf{ECRH.Equiv})$ *of efficient algorithms with the following syntax:*

$\mathsf{ECRH.Setup}(1^\lambda)$**:** *This is a probabilistic classical algorithm that takes the security parameter 1^λ as input and outputs a classical common reference string* crs*.*

$\mathsf{ECRH.Eval}(\mathsf{crs}, x)$**:** *This is a deterministic classical algorithm that takes a common reference string* crs *and an input $x \in \mathcal{X}$ as input and outputs a hash value $y \in \mathcal{Y}$.*

$\mathsf{ECRH.Gen}(\mathsf{crs})$**:** *This is a quantum algorithm that takes a common reference string* crs *as input, and outputs a hash value $y \in \mathcal{Y}$ and a quantum secret key $|\mathsf{sk}\rangle$.*

$\mathsf{ECRH.Equiv}^{|p\rangle}(1^t, |\mathsf{sk}\rangle, b)$ *This is a quantum algorithm that is given a quantumly-accessible classical oracle that computes a function $p : \mathcal{X} \to \{0,1\}$ and an "iteration parameter" 1^t, a secret key $|sk\rangle$, and a bit $b \in \{0,1\}$ as input and outputs $x \in \mathcal{X}$.*

As correctness, we require that for any $p : \mathcal{X} \to \{0,1\}$ and $t \in \mathbb{N}$, if we have

$$\Pr_{x \xleftarrow{\$} \mathcal{X}}[\mathsf{ECRH.Eval}(\mathsf{crs}, x) = y \ \wedge \ p(x) = b \mid \mathsf{ECRH.Eval}(\mathsf{crs}, x) = y] \geq t^{-1},$$

for all $\mathsf{crs} \in \mathsf{ECRH.Setup}(1^\lambda)$*, $y \in \mathcal{Y}$, and $b \in \{0,1\}$, then we have*

$$\Pr\left[\begin{array}{l|l} \mathsf{ECRH.Eval}(\mathsf{crs}, x) = y & \mathsf{crs} \xleftarrow{\$} \mathsf{ECRH.Setup}(1^\lambda), \\ \wedge\ p(x) = b & (y, |\mathsf{sk}\rangle) \xleftarrow{\$} \mathsf{ECRH.Gen}(\mathsf{crs}), \\ & x \xleftarrow{\$} \mathsf{ECRH.Equiv}^{|p\rangle}(1^t, |\mathsf{sk}\rangle, b) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

As security, we require that $\mathsf{ECRH.Eval}(\mathsf{crs}, \cdot)$ *is collision-resistant, i.e., for any efficient quantum adversary $\mathcal{A}$, we have*

$$\Pr\left[\begin{array}{l|l} \mathsf{ECRH.Eval}(\mathsf{crs}, x) = \mathsf{ECRH.Eval}(\mathsf{crs}, x') & \mathsf{crs} \xleftarrow{\$} \mathsf{ECRH.Setup}(1^\lambda), \\ \wedge\ x \neq x' & (x, x') \xleftarrow{\$} \mathcal{A}(\mathsf{crs}) \end{array}\right] = \mathsf{negl}(\lambda).$$

The above definition is similar to that of a family of equivocal collision-resistant hash functions in [AGKZ20], but stronger than that. The difference

is that the predicate p is specified by $\mathsf{ECRH.Gen}$ in the original definition (and $\mathsf{ECRH.Equiv}$ is not given oracle access to p and the iteration parameter 1^t since they can be hardwired into the algorithm) whereas we require the correctness for a general predicate p. They gave a construction of a family of equivocal collision resistant hash functions w.r.t. a predicate p that just returns the first bit of its input relative to a classical oracle. We observe that essentially the same construction actually works for general predicates. Thus, we obtain the following lemma.

Lemma 3.6. *There exists a family of equivocal collision resistant hash functions for general predicates with a domain $\{0,1\}^{2\lambda}$ and a range $\{0,1\}^{\lambda}$ relative to a classical oracle that is independent of the random oracle. In the construction, for any crs and y, we have*

$$\left|x \in \{0,1\}^{2\lambda} : \mathsf{ECRH.Eval}(\mathsf{crs}, x) = y\right| = 2^{\lambda}.$$

A proof of the above lemma can be found in the full version.

We construct a publicly verifiable PoQRO based on ECRH for the random oracle predicate.

Let $(\mathsf{ECRH.Setup}, \mathsf{ECRH.Gen}, \mathsf{ECRH.Eval}, \mathsf{ECRH.Equiv})$ be an ECRH for general predicates as in Lemma 3.6. Let $m = \omega(\log \lambda)$ be an integer. Let $H : \{0,1\}^{2\lambda} \to \{0,1\}$ and $H' : \{0,1\}^{2m\lambda} \to \{0,1\}^m$ be random oracles.[8] Then our publicly verifiable PoQRO is described as follows:

$\mathsf{PoQRO.Setup}(1^{\lambda})$**:** It generates $\mathsf{crs} \xleftarrow{\$} \mathsf{ECRH.Setup}(1^{\lambda})$ and outputs $\mathsf{pk} := \mathsf{crs}$.

$\mathsf{PoQRO.Prove}^{|H\rangle,|H'\rangle}(\mathsf{pk})$**:** It parses $\mathsf{crs} \leftarrow \mathsf{pk}$, computes $(y_i, |\mathsf{sk}_i\rangle) \xleftarrow{\$} \mathsf{ECRH.Gen}(\mathsf{crs})$ for all $i \in [m]$, $c := H'(y_1||...||y_m)$, $x_i \xleftarrow{\$} \mathsf{ECRH.Equiv}^{|H\rangle}(1^3, |\mathsf{sk}_i\rangle, c_i)$ for all $i \in [m]$ where c_i denotes the i-th bit of c, and outputs $\pi := \{(x_i, y_i)\}_{i\in[m]}$.

$\mathsf{PoQRO.Verify}^{H,H'}(\mathsf{pk}, \pi)$**:** It parses $\mathsf{crs} \leftarrow \mathsf{pk}$ and $\{(x_i, y_i)\}_{i\in[m]} \leftarrow \pi$ and outputs $\top$ if and only if $\mathsf{ECRH.Eval}(\mathsf{crs}, x_i) = y_i$ and $H(x_i) = c_i$ hold for all $i \in [m]$.

Lemma 3.7. *The above PoQRO satisfies correctness and soundness as required in Definition 3.1. Moreover, the construction is relativizing, i.e., that works relative to any oracles.*

Proof. (sketch) For any crs and y, since we assume

$$\left|x \in \{0,1\}^{2\lambda} : \mathsf{ECRH.Eval}(\mathsf{crs}, x) = y\right| = 2^{\lambda},$$

by the Chernoff bound, for an overwhelming fraction of H, we have

$$\Pr_{x \xleftarrow{\$} \{0,1\}^{2\lambda}}[\mathsf{ECRH.Eval}(\mathsf{crs}, x) = y \ \wedge \ H(x) = b \mid \mathsf{ECRH.Eval}(\mathsf{crs}, x) = y] \geq 1/3.$$

[8] Two (quantum) random oracles can be implemented by a single (quantum) random oracle by considering the first bit of the input as an index that specifies which random oracle to access.

Therefore, the correctness of the underlying ECRH immediately implies correctness of the above protocol.

Here, we only give a proof sketch for soundness. See the full version for a full proof. Roughly speaking, soundness can be proven as follows: First, we observe that the above protocol can be seen as a protocol obtained by applying Fiat-Shamir transform to a 4-round protocol where c is chosen by the verifier after receiving $\{y_i\}_{i\in[m]}$ from the prover. As shown in [LZ19b, DFMS19, DFM20], Fiat-Shamir transform preserves soundness even in the quantum setting.[9] Therefore, it suffices to prove soundness of the 4-round protocol against a cheating prover with classical access to the random oracle H. This can be argued as follows: Let $\{y_i\}_{i\in[m]}$ be the adversary's second message. and $\{x_i\}_{i\in[m]}$ be the fourth message. Without loss of generality, we assume that the adversary queries x_i for all $i \in [m]$ to the random oracle H and does not make the same query twice. By the collision-resistance of ECRH, the only preimage of y_i that is contained in the adversary's random oracle query list is x_i for all $i \in [m]$ with overwhelming probability. Conditioned on this, the adversary can win only if $H(x_i) = c_i$ holds for all $i \in [m]$, which happens with probability 2^{-m}. Therefore, the adversary can win with probability at most $2^{-m} + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$.

Finally, we remark that the above reduction works relative to any oracles. □

By combining Lemma 3.6 and 3.7, Theorem 3.4 follows.

3.2 Separations for Digital Signatures

In this section, we construct digital signature schemes that are secure in the ROM but insecure in the QROM based on PoQRO.

Lemma 3.8. *If there exist a PoQRO and a digital signature scheme that is EUF-CMA secure against quantum adversaries in the ROM, then there exists a digital signature scheme that is EUF-CMA secure in the ROM but not* 1*-EUF-CMA secure in the QROM.*

Lemma 3.9. *If there exist a publicly verifiable PoQRO and a digital signature scheme that is EUF-CMA secure against quantum adversaries in the ROM, then there exists a digital signature scheme that is EUF-CMA secure in the ROM but not EUF-NMA secure in the QROM.*

By combining the above lemmas with Theorem 3.3 and 3.4 and the fact that there exists a digital signature scheme that is EUF-CMA secure against quantum adversaries in the ROM under the QLWE assumption [GPV08], we obtain the following corollaries.

Corollary 3.10. *If the QLWE assumption holds, then there exists a digital signature scheme that is EUF-CMA secure against quantum adversaries in the ROM but not* 1*-EUF-CMA secure against quantum adversaries in the QROM.*

[9] Actually, since we only consider quantum adversaries that are only given *classical* access to the random oracle, there is a simpler analysis than those in [LZ19b, DFMS19, DFM20] as shown in the full version.

Corollary 3.11. *There exists a classical oracle relative to which there exists digital signature scheme that is EUF-CMA secure against quantum adversaries in the ROM but not EUF-NMA secure against quantum adversaries in the QROM.*[10]

We give proofs of Lemmas 3.8 and 3.9 in the following.

Proof. (of Lemma 3.8.) Let $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ be a digital signature scheme that is EUF-CMA secure against quantum adversaries in the ROM and $(\mathsf{PoQRO.Setup}, \mathsf{PoQRO.Prove}, \mathsf{PoQRO.Verify})$ be a PoQRO. Then we consider a digital signature scheme $(\mathsf{Sig.KeyGen}', \mathsf{Sig.Sign}', \mathsf{Sig.Verify}')$ that uses a random oracle H described below:[11]

$\mathsf{Sig.KeyGen}'^{H}(1^\lambda)$**:** This algorithm generates $(\mathsf{vk}, \mathsf{sigk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}^{H}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PoQRO.Setup}(1^\lambda)$, and outputs $\mathsf{vk}' := (\mathsf{vk}, \mathsf{pk})$ and $\mathsf{sigk}' := (\mathsf{sigk}, \mathsf{sk})$.

$\mathsf{Sig.Sign}'^{H}(\mathsf{sigk}' = (\mathsf{sigk}, \mathsf{sk}), m)$**:** If $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, m) = \top$, then it outputs sigk. Otherwise, it outputs $\sigma \xleftarrow{\$} \mathsf{Sig.Sign}^{H}(\mathsf{sigk}, m)$.

$\mathsf{Sig.Verify}'^{H}(\mathsf{vk}' = (\mathsf{vk}, \mathsf{pk}), m, \sigma)$**:** This algorithm works in exactly the same way as $\mathsf{Sig.Verify}^{H}(\mathsf{vk}, m, \sigma)$.

By the security of PoQRO, any efficient quantum adversary with *classical* access to H cannot find m such that $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, m) = \top$ with non-negligible probability. Therefore, we can reduce the EUF-CMA security of the above scheme against quantum adversaries in the ROM to that of the underlying scheme in the ROM in a straightforward manner.

On the other hand, a quantum polynomial-time adversary with *quantum* access to H can find m such that $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, m) = \top$ with overwhelming probability by correctness of PoQRO. Therfore, the adversary can obtain sigk by querying such an m to the signing oracle to obtain sigk. This enables the adversary to forge a signature on any message, and thus the above scheme is not 1-EUF-CMA secure against quantum adversaries in the QROM. □

Proof. (of Lemma 3.9.) Let $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ be a digital signature scheme that is EUF-CMA secure against quantum adversaries in the ROM and $(\mathsf{PoQRO.Setup}, \mathsf{PoQRO.Prove}, \mathsf{PoQRO.Verify})$ be a publicly verifiable PoQRO. Then we consider a digital signature scheme $(\mathsf{Sig.KeyGen}', \mathsf{Sig.Sign}', \mathsf{Sig.Verify}')$ that uses a random oracle H described below:

$\mathsf{Sig.KeyGen}'^{H}(1^\lambda)$**:** This algorithm generates $(\mathsf{vk}, \mathsf{sigk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}^{H}(1^\lambda)$ and $\mathsf{pk} \xleftarrow{\$} \mathsf{PoQRO.Setup}(1^\lambda)$, and outputs $\mathsf{vk}' := (\mathsf{vk}, \mathsf{pk})$ and $\mathsf{sigk}' := \mathsf{sigk}$.

[10] We do not need any computational assumption in this corollary since we can construct a EUF-CMA secure digital signature scheme relative to a classical oracle in a straightforward manner.

[11] We take the domain and range of H sufficiently largely so that this can be used for both PoQRO and the digital signature scheme. A similar remark also applies to constructions in proofs of Lemmas 3.9, 3.12, and 3.13.

$\mathsf{Sig.Sign}'^{H}(\mathsf{sigk}' = \mathsf{sigk}, m)$**:** This algorithm works in exactly the same way as $\mathsf{Sig.Sign}^{H}(\mathsf{sigk}, m)$.

$\mathsf{Sig.Verify}'^{H}(\mathsf{vk}' = (\mathsf{vk}, \mathsf{pk}), m, \sigma)$**:** This algorithm returns $\top$ if $\mathsf{Sig.Verify}^{H}(\mathsf{vk}, m, \sigma) = \top$ or $\mathsf{PoQRO.Verify}^{H}(\mathsf{pk}, m) = \top$.

By the security of PoQRO, any efficient quantum adversary with *classical* access to H cannot find m such that $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, m) = \top$ with non-negligible probability. Therefore, we can reduce the EUF-CMA security of the above scheme against quantum adversaries in the ROM to that of the underlying scheme in the ROM in a straightforward manner.

On the other hand, a quantum polynomial-time adversary with *quantum* access to H can find m such that $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, m) = \top$ with overwhelming probability by correctness of PoQRO. Then, the adversary can just output such an m and an arbitrary string σ as a forgery without making any signing query. Therefore, the above scheme is not EUF-NMA secure against quantum adversaries in the QROM. □

3.3 Separations for Public Key Encryption

In this section, we construct a PKE scheme schemes that are secure in the ROM but insecure in the QROM based on PoQRO.

Lemma 3.12. *If there exist a PoQRO and a PKE scheme that is IND-CCA secure against quantum adversaries in the ROM, then there exists a PKE scheme that is IND-CCA secure against quantum adversaries in the ROM but not* 1*-IND-CCA secure in the QROM.*

Lemma 3.13. *If there exist a publicly verifiable PoQRO and a PKE scheme that is IND-CCA secure against quantum adversaries in the ROM, then there exists a PKE scheme that is IND-CCA secure against quantum adversaries in the ROM but not IND-CPA secure in the QROM.*

By combining the above lemmas with Theorem 3.3 and 3.4 and the fact that there exists an IND-CCA secure PKE scheme in the standard model (and thus in the ROM) under the QLWE assumption [PW08], we obtain the following corollaries.

Corollary 3.14. *If the QLWE assumption holds, then there exists a PKE scheme that is IND-CCA secure against quantum adversaries in the ROM but not* 1*-IND-CCA secure in the QROM.*

Corollary 3.15. *There exists a classical oracle relative to which there exists a PKE scheme that is IND-CCA secure against quantum adversaries in the ROM but not IND-CPA secure in the QROM.*[12]

We give proofs of Lemmas 3.12 and 3.13 in the following.

[12] We do not need any computational assumption in this corollary since we can construct an IND-CCA secure PKE scheme relative to a classical oracle in a straightforward manner.

Proof. (of Lemma 3.12.) Let $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme that is IND-CCA secure against quantum adversaries in the ROM and $(\mathsf{PoQRO.Setup}, \mathsf{PoQRO.Prove}, \mathsf{PoQRO.Verify})$ be a PoQRO. Then we consider a PKE scheme $(\mathsf{PKE.KeyGen}', \mathsf{PKE.Enc}', \mathsf{PKE.Dec}')$ that uses a random oracle H described below:

$\mathsf{PKE.KeyGen}'^{H}(1^\lambda)$**:** This algorithm generates $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}^{H}(1^\lambda)$ and $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{PoQRO.Setup}(1^\lambda)$, and outputs $\mathsf{ek}' := (\mathsf{ek}, \mathsf{pk})$ and $\mathsf{dk}' := (\mathsf{dk}, \mathsf{sk})$.

$\mathsf{PKE.Enc}'^{H}(\mathsf{ek}' = (\mathsf{ek}, \mathsf{pk}), m)$**:** This algorithm works in exactly the same way as $\mathsf{PKE.Enc}^{H}(\mathsf{ek}, m)$.

$\mathsf{PKE.Dec}'^{H}(\mathsf{dk}' = (\mathsf{dk}, \mathsf{sk}), \mathsf{ct})$**:** If $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, \mathsf{ct}) = \top$, then it outputs dk. Otherwise, it outputs $m \xleftarrow{\$} \mathsf{PKE.Dec}^{H}(\mathsf{dk}, \mathsf{ct})$.

By the security of PoQRO, any quantum polynomial-time adversary with *classical* access to H cannot find ct such that $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, \mathsf{ct}) = \top$ with non-negligible probability. Therefore, we can reduce the CCA security of the above scheme against quantum adversaries in the ROM to that of the underlying scheme in a straightforward manner.

On the other hand, a quantum polynomial-time adversary with *quantum* access to H can find ct such that $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, \mathsf{ct}) = \top$ with overwhelming probability by correctness of PoQRO. Therfore, the adversary can obtain dk by querying such an ct to the decryption oracle to obtain dk. This enables the adversary to decrypt any ciphertext, and thus the above scheme is not 1-IND-CCA secure against quantum adversaries in the QROM. □

Proof. (of Lemma 3.13.) Let $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme with message space that is IND-CCA secure against quantum adversaries in the ROM and $(\mathsf{PoQRO.Setup}, \mathsf{PoQRO.Prove}, \mathsf{PoQRO.Verify})$ be a publicly verifiable PoQRO with proof space. We assume that a message of PKE scheme can be parsed as (m, π) where m is any bit-string and π is in the proof space of the PoQRO. Then we consider a PKE scheme $(\mathsf{PKE.KeyGen}', \mathsf{PKE.Enc}', \mathsf{PKE.Dec}')$ that uses a random oracle H described below:

$\mathsf{PKE.KeyGen}'^{H}(1^\lambda)$**:** This algorithm generates $(\mathsf{ek}, \mathsf{dk}) \xleftarrow{\$} \mathsf{PKE.KeyGen}^{H}(1^\lambda)$ and $\mathsf{pk} \xleftarrow{\$} \mathsf{PoQRO.Setup}(1^\lambda)$, and outputs $\mathsf{ek}' := (\mathsf{ek}, \mathsf{pk})$ and $\mathsf{dk}' := \mathsf{sk}$.

$\mathsf{PKE.Enc}'^{H}(\mathsf{ek}' = (\mathsf{ek}, \mathsf{pk}), m' = (m, \pi))$**:** If $\mathsf{PoQRO.Verify}^{H}(\mathsf{pk}, \pi) = \top$, then it outputs $\mathsf{ct} := m$. Otherwise, it outputs $\mathsf{ct} \xleftarrow{\$} \mathsf{PKE.Enc}^{H}(\mathsf{ek}, m')$.

$\mathsf{PKE.Dec}'^{H}(\mathsf{dk}' = \mathsf{dk}, \mathsf{ct})$**:** This algorithm works in exactly the same way as $\mathsf{PKE.Dec}^{H}(\mathsf{dk}, \mathsf{ct})$.

By the security of PoQRO, any quantum polynomial-time adversary with *classical* access to H cannot find π such that $\mathsf{PoQRO.Verify}^{H}(\mathsf{sk}, \mathsf{ct}) = \top$ with non-negligible probability. Therefore, we can reduce the IND-CPA security of the above scheme against quantum adversaries in the ROM to that of the underlying scheme in a straightforward manner.

On the other hand, a quantum polynomial-time adversary with *quantum* access to H can find π such that $\mathsf{PoQRO.Verify}^H(\mathsf{pk}, \pi) = \top$ with overwhelming probability by correctness of PoQRO. Therfore, if the adversary query $(m'_0 := (m_0, \pi), m'_1 := (m_1, \pi))$ for $m_0 \neq m_1$ as a challenge query, then the challenge ciphertext is equal to m'_0 or m'_1, and thus it can trivially break the IND-CPA security. Therefore, the above scheme is not IND-CPA secure against quantum adversaries in the QROM. □

4 Lifting Theorem

In this section, we prove a lifting theorem from ROM security to QROM security for a certain type of security notions. Then we discuss applications of this theorem.

4.1 Statement of Lifting Theorem

First, we define a concept of classically verifiable games. The following formalization is based on the definition of falsifiable assumptions in [GW11].

Definition 4.1 (Classically verifiable games.)**.** *A classically verifiable game consists of an interactive classical challenger $\mathcal{C}^H$ that is given classical access to a random oracle H and a constant $c \in [0, 1)$. In the ROM (resp. QROM), the challenger $\mathcal{C}^H(1^\lambda)$ interacts with an adversary $\mathcal{A}^H(1^\lambda)$ (resp. $\mathcal{A}^{|H\rangle}(1^\lambda)$) and finally outputs $\top$ indicating acceptance or $\bot$ indicating rejection. If the challenger returns $\top$, we say that $\mathcal{A}^H(1^\lambda)$ (resp. $\mathcal{A}^{|H\rangle}(1^\lambda)$) wins $\mathcal{C}^H(1^\lambda)$.*

We say that a classically verifiable game is hard in the ROM (resp. QROM) if for any efficient quantum[13] adversary $\mathcal{A}^H$ (resp. $\mathcal{A}^{|H\rangle}$) that is given a classical (resp. quantum) access to the random oracle H, we have

$$\Pr_H[\mathcal{A}^H(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)] \leq c + \mathsf{negl}(\lambda)$$
$$(resp. \Pr_H[\mathcal{A}^{|H\rangle}(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)] \leq c + \mathsf{negl}(\lambda))$$

where the probability is over the choice of the random oracle H, the random coins of $\mathcal{A}$ and $\mathcal{C}$, and the randomness in measurements by $\mathcal{A}$.[14]

We say that a classically verifiable game is search-type if $c = 0$.

Remark 2. *Though the above definition is based on the definition of falsifiable assumptions in [GW11], the hardness of a classically verifiable game may not be falsifiable since we allow the challenger to run in unbounded time.*

[13] Note that we consider quantum adversaries even in the classical ROM.

[14] We only write H in the subscript of the probability since all the other randomness are always in the probability space whenever we write a probability throughout this section.

Examples. Soundness of PoQRO can be seen as hardness of a search-type classically verifiable game in the ROM. On the other hand, completeness requires (at least) that the game is not hard in the QROM. Therefore, the existence of PoQRO implies 2-round search-type classically falsifiable cryptographic game that is hard in ROM but is not hard in QROM.

EUF-CMA and EUF-NMA security of digital signatures in the ROM (resp. QROM) require hardness of search-type classically falsifiable games in the ROM (resp. QROM).

CPA and CCA security of PKE in the ROM (resp. QROM) require hardness of classically falsifiable games in the ROM (resp. QROM), which are not search-type.

Our main lifting theorem is stated as follows.

Theorem 4.2 (Lifting Theorem for Search-Type Games)**.** *Let $\mathcal{C}$ be an k-classical-query challenger of a search-type classically verifiable game and $\mathcal{A}$ be a q-quantum-query efficient adversary against the game in the QROM. Then there exists a k-classical-query efficient adversary $\mathcal{B}$ against the game in the ROM such that*

$$\Pr_H[\mathcal{B}^H(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)] \geq \frac{1}{(2q+1)^{2k}} \Pr_H[\mathcal{A}^{|H\rangle}(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)].$$

In particular, for any search-type classically verifiable game in which the challenger makes at most $O(1)$ queries, if the game is hard in the ROM, then that is also hard in the QROM.

We also give a variant of the above theorem, which gives a slightly stronger inequality assuming that $\mathcal{C}$'s queries are publicly computable. Looking ahead, this variant will be used in Sec. 4.5 where we give quantum query lower bounds.

Theorem 4.3 (Lifting Theorem for Public-Query Search-Type Games)**.** *Let $\mathcal{C}$ and $\mathcal{A}$ be as in Theorem 4.2. Moreover, we assume that the game is public-query, i.e., the list of $\mathcal{C}$'s queries is determined by the transcript and computable in quantum polynomial-time. Then there exists a k-classical-query efficient adversary $\mathcal{B}$ against the game in the ROM such that*

$$\Pr_H[\mathcal{B}^H(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda) \wedge\ L_\mathcal{B} = L_\mathcal{C}] \geq \frac{1}{(2q+1)^{2k}} \Pr_H[\mathcal{A}^{|H\rangle}(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)].$$

where $L_\mathcal{B}$ and $L_\mathcal{C}$ are the list of random oracle queries by $\mathcal{B}$ and $\mathcal{C}$, respectively.

4.2 Proof of Lifting Theorem

For proving Theorem 4.2 and 4.3, we introduce a lemma from [DFM20]. For stating the lemma, we introduce some notations. Before giving formal definitions, we give a rough explanations. 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_j on j-th measured query. Formal definitions are given below:

Definition 4.4 (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 a function computed by $\mathcal{O}$ before the update. This updated oracle is used in the rest of execution of $\mathcal{A}$. We denote $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x, y)$ to mean the above reprogramming.

Definition 4.5 (Measure-and-Reprogram)**.** *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:*

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

1. *For each $j \in [k]$, uniformly pick $(i_j, b_j) \in ([q] \times \{0, 1\}) \cup \{(\bot, \bot)\}$ such that there does not exist $j \neq j'$ such that $i_j = i_{j'} \neq \bot$.*
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 i-th query, the oracle is simulated as follows:*
   (a) *If $i = i_j$ for some $j \in [k]$, measure $\mathcal{A}$'s query register to obtain x'_j, and do either of the following.*
      i. *If $b_j = 0$, reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_j, y_j)$ and answer $\mathcal{A}$'s i_j-th query by using the reprogrammed oracle.*
      ii. *If $b_j = 1$, answer $\mathcal{A}$'s i_j-th query by using the oracle before the reprogramming and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_j, y_j)$.*
   (b) *Otherwise, answer $\mathcal{A}$'s i-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 $j \in [k]$ such that $i_j = \bot$, set $x'_j := x_j$.*
5. *Output $\mathbf{x}' := ((x'_1, ..., x'_k), z)$.*

Then we state [DFM20, Theorem 6] with alternative notations as defined above.

Lemma 4.6. *(Rephrasing of [DFM20, Theorem 6]) Let $\mathcal{X}$, $\mathcal{Y}$, $\mathcal{Z}$, and $\mathcal{A}$ be as in Definition 4.5. Then for any* inp, $H : \mathcal{X} \to \mathcal{Y}$, $\mathbf{x}^* = (x_1^*, ..., x_k^*) \in \mathcal{X}^k$ *such that $x_j^* \neq x_{j'}^*$ for all $j \neq j'$, $\mathbf{y} = (y_1, ..., y_k) \in \mathcal{Y}^k$, and a relation $R \subseteq \mathcal{X}^k \times \mathcal{Y}^k \times \mathcal{Z}$, we have*

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

where $\tilde{A}[H, \mathbf{y}]$ is the measure-and-reprogram algorithm as defined in Definition 4.5 and $H_{\mathbf{x}^*, \mathbf{y}}$ is defined as

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

We prove Theorem 4.2 by using Lemma 4.6.

Proof. (of Theorem 4.2.) We prove a slightly stronger claim than Theorem 4.2, where we switch the order of the quantifiers of $\mathcal{B}$ and $\mathcal{C}$. Specifically, we prove that for any q-quantum-query efficient algorithm $\mathcal{A}$, there exists an k-classical-query efficient algorithm $\mathcal{B}$ such that for any k-classical-query challenger $\mathcal{C}$, we have

$$\Pr_H[\mathcal{B}^H(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)] \geq \frac{1}{(2q+1)^{2k}} \Pr_H[\mathcal{A}^{|H\rangle}(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)]. \tag{2}$$

For proving this claim, it suffices to prove it assuming $\mathcal{C}$ is deterministic since the claim for probabilistic $\mathcal{C}$ immediately follows from that for deterministic $\mathcal{C}$ by a simple averaging argument.[15] Therefore, in the following, we assume that $\mathcal{C}$ is deterministic. We also assume that $\mathcal{C}$ does not make the same query twice and makes exactly k queries (by introducing dummy queries if necessary) without loss of generality.

We construct $\mathcal{B}$ as follows:

$\mathcal{B}^H(1^\lambda)$**:** This is an algorithm that interacts with a challenger as follows:

1. Chooses a function $H' : \mathcal{X} \to \mathcal{Y}$ from a family of $2q$-wise independent hash functions.
2. For each $j \in [k]$, uniformly pick $(i_j, b_j) \in ([q] \times \{0, 1\}) \cup \{(\bot, \bot)\}$ so that there does not exist $j \neq j'$ such that $i_j = i_{j'} \neq \bot$.

[15] Here, it is important that $\mathcal{B}$ does not depend on $\mathcal{C}$ due to the switching of the order of quantifiers.

3. Run $\mathcal{A}^{\mathcal{O}}(1^\lambda)$ by forwarding all messages supposed to be sent to the challenger to the external challenger and forwarding all messages sent back from the external challenger to $\mathcal{A}$ and simulating the oracle $\mathcal{O}$ as follows. Initialize $\mathcal{O}$ to be a quantumly-accessible classical oracle that computes H'. When $\mathcal{A}$ makes its i-th query, the oracle is simulated as follows:
   (a) If $i = i_j$ for some $j \in [k]$, measure $\mathcal{A}$'s query register to obtain x'_j, query x'_j to the random oracle H to obtain $H(x'_j)$, and do either of the following.
      i. If $b_j = 0$, reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_j, H(x'_j))$ and answer $\mathcal{A}$'s i_j-th query by using the reprogrammed oracle.
      ii. If $b_j = 1$, answer $\mathcal{A}$'s i_j-th query by using the oracle before the reprogramming and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_j, H(x'_j))$.
   (b) Otherwise, answer $\mathcal{A}$'s i-th query by just using the oracle $\mathcal{O}$ without any measurement or reprogramming.

It is clear that $\mathcal{B}$ only makes k classical queries to H and is efficient if $\mathcal{A}$ is efficient. We prove that $\mathcal{B}$ satisfies Eq. 2 for all k-classical-query challengers $\mathcal{C}$. Let $\mathcal{X}$ and $\mathcal{Y}$ be the domain and codomain of a random oracle that is used in the game, and $\mathcal{Z}$ be a set consisting of all possible transcripts between $\mathcal{A}$ and $\mathcal{C}$. Here, a transcript means a concatenation of all messages exchanged between $\mathcal{A}$ and $\mathcal{C}$ and does not contain query-response pairs of the oracle. We call the concatenation of all query-response pairs for $\mathcal{C}$ and the transcript a $\mathcal{C}$'s view. We denote $\mathcal{C}$'s view in the form of $(\mathbf{x} = (x_1, ..., x_k), \mathbf{y} = (y_1, ..., y_k), z) \in \mathcal{X}^k \times \mathcal{Y}^k \times \mathcal{Z}$ where (x_j, y_j) is the j-th query-response pair for $\mathcal{C}$ and z is the transcript. Since we assume that $\mathcal{C}$ is deterministic, a view determines if $\mathcal{C}$ accepts or rejects. Let $R_\lambda \subseteq \mathcal{X}^k \times \mathcal{Y}^k \times \mathcal{Z}$ be a relation consisting of accepting view with respect to the security parameter λ. More precisely, for $(\mathbf{x} = (x_1, ..., x_k), \mathbf{y} = (y_1, ..., y_k), z) \in \mathcal{X}^k \times \mathcal{Y}^k \times \mathcal{Z}$, $(\mathbf{x}, \mathbf{y}, z) \in R_\lambda$ if the following algorithm $\mathsf{VerView}$ returns $\top$ on input $(1^\lambda, \mathbf{x}, \mathbf{y}, z)$.

$\mathsf{VerView}(1^\lambda, \mathbf{x} = (x_1, ..., x_k), \mathbf{y} = (y_1, ..., y_k), z)$**:** Run $\mathcal{C}(1^\lambda)$ by simulating all messages supposed to be sent from $\mathcal{A}$ and random oracle's responses so that they are consistent with the view $(\mathbf{x}, \mathbf{y}, z)$. At some point in the simulation, if $\mathcal{C}$'s behavior is not consistent with the view (i.e., $\mathcal{C}$ sends a message that is not consistent with the transcript z or its j-th query is not equal to x_j), then $\mathsf{VerView}$ returns $\bot$. Otherwise, $\mathsf{VerView}$ outputs the final output of $\mathcal{C}$.

We remark that $\mathsf{VerView}$ is deterministic as we assume $\mathcal{C}$ is deterministic and thus the relation R_λ is well-defined.

For a function $H : \mathcal{X} \to \mathcal{Y}$, we consider a quantum algorithm $\mathcal{S}_H$, in which the function H is hardwired, that is given quantum access to an oracle that computes another function $H' : \mathcal{X} \to \mathcal{Y}$ described as follows:

$\mathcal{S}_H^{|H'\rangle}(1^\lambda)$**:** Simulate an interaction between $\mathcal{A}$ and $\mathcal{C}$ by simulating oracles for them as follows:

- $\mathcal{A}$'s queries are just forwarded to the oracle $|H'\rangle$ and responded as $|H'\rangle$ responds.
- For $\mathcal{C}$'s j-th query x_j for $j \in [k]$, the oracle returns $H(x_j)$.

Finally, it outputs $\mathcal{C}$'s queries $\mathbf{x} := (x_1, ..., x_k)$ and the transcript z between $\mathcal{A}$ and $\mathcal{C}$ in the above execution.

For any $\lambda \in \mathbb{N}$, $H, H' : \mathcal{X} \to \mathcal{Y}$, $\mathbf{x}^* = (x_1^*, ..., x_k^*) \in \mathcal{X}^k$ such that $x_j^* \neq x_{j'}^*$ for all $j \neq j'$, and $\mathbf{y} = (y_1, ..., y_k) \in \mathcal{Y}^k$, by applying Lemma 4.6 for $\mathcal{S}_H$, we have

$$\begin{aligned}&\Pr[\mathbf{x}' = \mathbf{x}^* \wedge (\mathbf{x}', \mathbf{y}, z) \in R_\lambda : (\mathbf{x}', z) \xleftarrow{\$} \tilde{\mathcal{S}}_H[H', \mathbf{y}](1^\lambda)] \\ &\quad \geq \frac{1}{(2q+1)^{2k}} \Pr[\mathbf{x} = \mathbf{x}^* \wedge (\mathbf{x}, \mathbf{y}, z) \in R_\lambda : (\mathbf{x}, z) \xleftarrow{\$} \mathcal{S}_H^{|H'_{\mathbf{x}^*, \mathbf{y}}\rangle}(1^\lambda)].\end{aligned} \tag{3}$$

where $\tilde{\mathcal{S}}_H[H', \mathbf{y}]$ is to $\mathcal{S}_H$ as $\tilde{\mathcal{A}}[H', \mathbf{y}]$ is to $\mathcal{A}$ as defined in Definition 4.5 and $H'_{\mathbf{x}^*, \mathbf{y}}$ is as defined in Lemma 4.6.

Especially, since the above inequality holds for any $\mathbf{y}$, by setting $\mathbf{y} := H(\mathbf{x}^*) = (H(x_1^*), ..., H(x_k^*))$, we have

$$\begin{aligned}&\Pr[\mathbf{x}' = \mathbf{x}^* \wedge (\mathbf{x}', H(\mathbf{x}^*), z) \in R_\lambda : (\mathbf{x}', z) \xleftarrow{\$} \tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)] \\ &\quad \geq \frac{1}{(2q+1)^{2k}} \Pr[\mathbf{x} = \mathbf{x}^* \wedge (\mathbf{x}, H(\mathbf{x}^*), z) \in R_\lambda : (\mathbf{x}, z) \xleftarrow{\$} \mathcal{S}_H^{|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle}(1^\lambda)].\end{aligned} \tag{4}$$

Recall that $\mathcal{S}_H^{|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle}(1^\lambda)$ is an algorithm that simulates an interaction between $\mathcal{A}$ and $\mathcal{C}$ where $\mathcal{A}$'s oracle and $\mathcal{C}$'s oracles are simulated by $|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle$ and H, respectively, and outputs $\mathcal{C}$'s queries $\mathbf{x}$ and the transcript z. Thus, conditioned on that $\mathbf{x} = \mathbf{x}^*$, $\mathcal{S}_H^{|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle}(1^\lambda)$ simulates an interaction between $\mathcal{A}$ and $\mathcal{C}$ where both oracles of $\mathcal{A}$ and $\mathcal{C}$ compute the same function $H'_{\mathbf{x}^*, H(\mathbf{x}^*)}$ since we have $H(\mathbf{x}^*) = H'_{\mathbf{x}^*, H(\mathbf{x}^*)}(\mathbf{x}^*)$ by definition. Moreover, conditioned on that $\mathbf{x} = \mathbf{x}^*$, $(\mathbf{x}, H(\mathbf{x}^*), z) \in R_\lambda$ is equivalent to that $\mathcal{A}^{|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle}(1^\lambda)$ wins $\mathcal{C}^{H'_{\mathbf{x}^*, H(\mathbf{x}^*)}}(1^\lambda)$ in the execution simulated by $\mathcal{S}_H^{|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle}(1^\lambda)$. Based on these observations, we have

$$\begin{aligned}&\Pr[\mathbf{x} = \mathbf{x}^* \wedge (\mathbf{x}, H(\mathbf{x}^*), z) \in R_\lambda : (\mathbf{x}, z) \xleftarrow{\$} \mathcal{S}_H^{|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle}(1^\lambda)] \\ &= \Pr[\mathbf{x} = \mathbf{x}^* \wedge \mathcal{A}^{|H'_{\mathbf{x}^*, H(\mathbf{x}^*)}\rangle}(1^\lambda) \text{ wins } \mathcal{C}^{H'_{\mathbf{x}^*, H(\mathbf{x}^*)}}(1^\lambda)]\end{aligned} \tag{5}$$

where $\mathbf{x}$ in the RHS is the list of queries made by $\mathcal{C}$.

Moreover, if we uniformly choose $H, H' : \mathcal{X} \to \mathcal{Y}$, then the distribution of the function $H'_{\mathbf{x}^*, H(\mathbf{x}^*)}$ is uniform over all functions from $\mathcal{X}$ to $\mathcal{Y}$ for any fixed $\mathbf{x}^*$. Therefore, by substituting Eq. 5 for the RHS of Eq. 4, taking the average over the random choice of H and H', and summing up over all $\mathbf{x}^* \in \mathcal{X}^k$, we

have

$$\sum_{\mathbf{x}^*\in\mathcal{X}^k} \Pr_{H,H'}[\mathbf{x}'=\mathbf{x}^* \wedge (\mathbf{x}', H(\mathbf{x}^*), z) \in R_\lambda : (\mathbf{x}', z) \xleftarrow{\$} \tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)] \\ \geq \frac{1}{(2q+1)^{2k}} \Pr_H[\mathcal{A}^{|H\rangle}(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)]. \tag{6}$$

For proving Eq. 2 and completing the proof, what is left is to prove that the LHS of Eq. 6 is smaller than or equal to the LHS of Eq. 2. For proving this, we spell out how $\tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)]$ works according to the definition:

$\tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)$**:** Given the security parameter 1^λ as input, it works as follows:

1. For each $j \in [k]$, uniformly pick $(i_j, b_j) \in ([q] \times \{0,1\}) \cup \{(\bot, \bot)\}$ so that there does not exist $j \neq j'$ such that $i_j = i_{j'} \neq \bot$.
2. Simulate the interaction between $\mathcal{A}$ and $\mathcal{C}$ by simulating oracles for them as follows:

   Initialize an oracle $\mathcal{O}$ to be a quantumly-accessible classical oracle that computes H'. When $\mathcal{A}$ makes its i-th query, the oracle is simulated as follows:

   (a) If $i = i_j$ for some $j \in [k]$, measure $\mathcal{A}$'s query register to obtain x'_j, and do either of the following.
      i. If $b_j = 0$, reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_j, H(x^*_j))$ and answer $\mathcal{A}$'s i_j-th query by using the reprogrammed oracle.
      ii. If $b_j = 1$, answer $\mathcal{A}$'s i_j-th query by using the oracle before the reprogramming and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_j, H(x^*_j))$.

   (b) Otherwise, answer $\mathcal{A}$'s i-th query by just using the oracle $\mathcal{O}$ without any measurement or reprogramming.

   When $\mathcal{C}$ makes its j-th query x_j, return $H(x_j)$ as a response by the random oracle for each $j \in [k]$.

   Let z be the transcript in the above simulated execution.
3. For all $j \in [k]$ such that $i_j = \bot$, set $x'_j := x_j$.
4. Output $\mathbf{x}' := ((x'_1, ..., x'_k), z)$.

One can see from the above description that an execution of the game simulated by $\tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)$ for a randomly chosen H' is very close to an interaction between $\mathcal{B}^H$ and $\mathcal{C}^H$. The only difference is that $\mathcal{B}^H$ reprograms $\mathcal{O}$ to output $H(x'_j)$ instead of $H(x^*_j)$ on input x'_j in Step 2a.[16] Therefore, conditioned on that $\mathbf{x}' = \mathbf{x}^*$, $\tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)$ for a randomly chosen H' perfectly simulates an interaction between $\mathcal{B}^H$ and $\mathcal{C}^H$. Moreover, if $\mathbf{x}' = \mathbf{x}^*$ and $(\mathbf{x}', H(\mathbf{x}^*), z) \in R_\lambda$,

[16] Strictly speaking, there is another difference that we consider $\tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)$ for a uniformly chosen H' whereas $\mathcal{B}$ chooses H' from a family of $2q$-wise independent hash functions. However, by Lemma 2.1, this does not cause any difference.

then we must have $\mathbf{x} = \mathbf{x}^*$ where $\mathbf{x}$ is the list of $\mathcal{C}$'s queries in the simulation since otherwise the view $(\mathbf{x}', H(\mathbf{x}^*), z)$ is not consistent with $\mathcal{C}$'s queries and cannot pass $\mathsf{VerfView}$. In this case, we have $(\mathbf{x}, H(\mathbf{x}), z) \in R_\lambda$, which means that $\mathcal{B}^H$ wins $\mathcal{C}^H$ in the simulated execution. Therefore, for any fixed H and $\mathbf{x}^*$, we have

$$\begin{aligned}&\Pr_{H'}[\mathbf{x}' = \mathbf{x}^* \wedge (\mathbf{x}', H'(\mathbf{x}^*), z) \in R_\lambda : (\mathbf{x}', z) \xleftarrow{\$} \tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)] \\ \leq &\Pr[\mathbf{x} = \mathbf{x}^* \wedge \mathcal{B}^H(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)]\end{aligned} \tag{7}$$

where $\mathbf{x}$ in the RHS is the list of queries by $\mathcal{C}^H$.

By substituting Eq 7 for the LHS of Eq. 6, we obtain Eq. 2. This completes the proof of Theorem 4.2. □

Theorem 4.3 can be proven by a slight modification to the proof of Theorem 4.2.

Proof. (of Theorem 4.3.) We consider an algorithm $\mathcal{B}$ that works similarly to that in the proof of Theorem 4.2 except that it does an additional step at the end:

$\mathcal{B}^H(1^\lambda)$: This is an algorithm that interacts with a challenger as follows:

1-3. Work similarly to $\mathcal{B}$ in the proof of Theorem 4.2.

4. After completing the interaction with $\mathcal{C}$, compute the list of $\mathcal{C}$'s query, and if any query in the list has not yet been queried in the previous steps, then query them to H.

We have Eq. 6 by exactly the same argument to that in the proof of Theorem 4.2 since we do not use anything about the construction of $\mathcal{B}$ until this point. By the modification of $\mathcal{B}$ as described above, in the simulation of an interaction between $\mathcal{B}$ and $\mathcal{C}$ by $\tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)$, $\mathcal{B}$'s query list exactly matches $\mathbf{x}'$ that appears in the description of $\tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)$. With this observation in mind, by a similar argument to that in the proof of Theorem 4.2, we can see that we have

$$\begin{aligned}&\Pr_{H'}[\mathbf{x}' = \mathbf{x}^* \wedge (\mathbf{x}', H'(\mathbf{x}^*), z) \in R_\lambda : (\mathbf{x}', z) \xleftarrow{\$} \tilde{\mathcal{S}}_H[H', H(\mathbf{x}^*)](1^\lambda)] \\ \leq &\Pr[L_{\mathcal{B}} = L_{\mathcal{C}} = \{x_1^*, ..., x_k^*\} \wedge \mathcal{B}^H(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)]\end{aligned}. \tag{8}$$

By substituting Eq. 8 for the LHS of Eq. 6, we obtain

$$\Pr_H[\mathcal{B}^H(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda) \wedge L_{\mathcal{B}} = L_{\mathcal{C}}] \geq \frac{1}{(2q+1)^{2k}} \Pr_H[\mathcal{A}^{|H\rangle}(1^\lambda) \text{ wins } \mathcal{C}^H(1^\lambda)].$$

which completes the proof of Theorem 4.3. □

4.3 Immediate Corollaries

Here, we list immediate corollaries of Theorem 4.2.

PoQRO. Soundness of PoQRO can be seen as hardness of a search-type classically verifiable game in the ROM. On the other hand, completeness requires (at least) that the game is not hard in the QROM. By Theorem 4.2, such a separation between ROM and QROM is impossible if the number of verifier's query is $O(1)$. Therefore, we obtain the following corollary:

Corollary 4.7. *There does not exist PoQRO where the verification algorithm makes a constant number of random oracle queries.*

We note that a similar statement holds even for an interactive version of PoQRO.

(Non-)Interactive Arguments. A post-quantum interactive argument for a language L is a protocol where an efficient classical prover given a statement x and some auxiliary information (e.g., witness in the case of L is an NP language) and a efficient classical verifier only given x interacts and the verifier finally returns $\top$ indicating acceptance or $\bot$ indicating rejection. As correctness, we require that the verifier returns $\top$ with overwhelming probability if both parties run honestly. As (post-quantum) soundness, we require that any efficient cheating prover cannot let the verifier accept on any $x \notin L$ with a non-negligible probability.

Here, we consider constructions of interactive arguments based on random oracles. Clearly, soundness requirement of interactive arguments is captured by a search-type classically verifiable game. Therefore, by Theorem 4.2, we obtain the following corollary:

Corollary 4.8. *If an interactive argument with constant-query verifier is sound in the ROM, then it is also sound in the QROM.*

Non-interactive arguments (in the common reference string model) is defined similarly except that a common reference string is generated by a trusted third party and distributed to both the prover and the verifier at the beginning of the protocol and then the protocol consists of only one-round communication, i.e., a prover just sends a proof to the verifier and verifies it. (Adaptive) soundness of non-interactive arguments is defined similarly to soundness of interactive arguments with the modification that the statement $x \notin L$ for which the cheating prover tries to generate a forged proof can be chosen after seeing the common reference string.

Similarly, by Theorem 4.2, we obtain the following corollary:[17]

[17] Note that the theorem is applicable even though the soundness game for non-interactive arguments is not falsifiable since the challenger in our definition of classically verifiable games is not computationally bounded.

Corollary 4.9. *If a non-interactive argument is sound in the ROM with constant-query verifier and constant-query common reference string generation algorithm is sound in the ROM, then it is also sound in the QROM.*

Digital Signatures. As already observed, EUF-CMA security can be seen as a hardness of a search-type classically verifiable game. Therefore, as an immediate corollary of Theorem 4.2, we obtain the following corollary.

Corollary 4.10. *If a digital signature scheme is n-EUF-CMA secure in the ROM for $n = O(1)$ and the key generation, signing, and verification algorithms make $O(1)$ random oracle queries, then the scheme is also n-EUF-CMA secure in the QROM. If $n = 0$ (i.e., if we consider EUF-NMA security), then a similar statement holds even if the signing algorithm makes arbitrarily many queries.*

Unfortunately, we cannot extend this result to the ordinary EUF-CMA security where the number of signing query is unbounded (except for a non-interesting case where the signing algorithm does not make a random oracle query) since the challenger in the EUF-CMA game may make as many random oracle queries as the adversary's signing queries, which is not bounded by a constant. In Sec. 4.4, we extend the above corollary to give a lifting theorem for EUF-CMA security (without restricting the number of signing queries) assuming a certain structure for the scheme.

4.4 Application to Digital Signatures

Here, we discuss implications of our lifting theorem for digital signatures.

Theorem 4.11. *Suppose that a digital signature scheme* $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ *with a message space $\mathcal{M}$ relative to a random oracle $H : \mathcal{X} \to \mathcal{Y}$ is EUF-NMA secure against quantum adversaries in the ROM and satisfies the following properties:*

1. $\mathsf{Sig.KeyGen}$ *does not make a random oracle query and* $\mathsf{Sig.Verify}$ *makes $O(1)$ random oracle queries. (There is no restriction on the number of random oracle queries by* $\mathsf{Sig.Sign}$*.)*

2. *A random query made by* $\mathsf{Sig.Sign}$ *or* $\mathsf{Sig.Verify}$ *reveals the message given to them as input. More precisely, there exists a classically efficiently computable function $\mathsf{XtoM} : \mathcal{X} \to \mathcal{M}$ such that for any H, honestly generated $(\mathsf{vk}, \mathsf{sigk})$, m, and σ, if $\mathsf{Sig.Sign}^H(\mathsf{sk}, m)$ or $\mathsf{Sig.Verify}^H(\mathsf{vk}, m, \sigma)$ makes a random oracle query x, then we have $\mathsf{XtoM}(x) = m$.*

3. *A signature is simulatable without a signing key if we can (non-adaptively) program the random oracle. More precisely, there exist a classically efficiently computable function $F_{\mathsf{vk}} : \mathcal{R} \to \mathcal{Y}$ tagged by a verification key vk and an efficient classical algorithm $\mathcal{S}$ such that for any honestly generated*

$(\mathsf{vk}, \mathsf{sigk})$ *and* $m_1, ..., m_\ell$ *for* $\ell = \mathsf{poly}(\lambda)$*, we have*

$$\left\{ \left(\{H(x)\}_{x\in\mathcal{X}}, \{\sigma_i\}_{i\in[\ell]}\right) : \begin{array}{l} H \xleftarrow{\$} \mathsf{Func}(\mathcal{X},\mathcal{Y}) \\ \sigma_i \xleftarrow{\$} \mathsf{Sig.Sign}^H(\mathsf{sigk}, m_i) \text{ for all } i \in [\ell] \end{array} \right\}$$
$$\approx \left\{ \left(\{F_{\mathsf{vk}}(\widetilde{H}(x))\}_{x\in\mathcal{X}}, \{\sigma_i\}_{i\in[\ell]}\right) : \begin{array}{l} \widetilde{H} \xleftarrow{\$} \mathsf{Func}(\mathcal{X},\mathcal{R}) \\ \{\sigma_i\}_{i\in[\ell]} \xleftarrow{\$} \mathcal{S}^{\widetilde{H}}(\mathsf{vk}, m) \end{array} \right\}.$$

where $\approx$ *means that two distributions are statistically indistinguishable.*

Then the scheme is EUF-CMA secure against quantum adversaries in the QROM.

Examples. Though the requirements in the above theorem may seem quite restrictive, it captures at least two important constructions of digital signatures: FDH signatures (and its lattice-based variant by Gentry, Peikert, and Vaikuntanathan [GPV08]) and Fiat-Shamir signatures. See the full version for details.

Proof. (of Theorem 4.11.) Let $(\mathsf{Sig.KeyGen}, \mathsf{Sig.Sign}, \mathsf{Sig.Verify})$ be a digital signature scheme relative to a random oracle $H : \mathcal{X} \to \mathcal{Y}$ that satisfies conditions given in Theorem 4.11. By Corollary 4.10 and condition 1 of Theorem 4.11, the scheme is EUF-NMA secure against quantum adversaries in the QROM. Therefore, it suffices to prove that the EUF-NMA security implies the EUF-CMA security for a scheme that satisfies the above conditions.

Let $\mathcal{A}$ be an adversary against the EUF-CMA security in the QROM that makes at most q_H random oracle queries and q_{sig} signing queries. We consider the following sequence of experiments. We denote by T_i the event that Exp_i returns 1.

Exp_1**:** This is the original EUF-CMA experiment. That is, the challenger generates $H \xleftarrow{\$} \mathsf{Func}(\mathcal{X},\mathcal{Y})$ and $(\mathsf{vk}, \mathsf{sigk}) \xleftarrow{\$} \mathsf{Sig.KeyGen}(1^\lambda)$, the adversary $\mathcal{A}$ is given vk, quantum access to the random oracle that computes H and classical access to the signing oracle $\mathsf{Sig.Sign}^H(\mathsf{sigk}, \cdot)$, and $\mathcal{A}$ finally returns (m^*, σ^*). The experiment returns 1 if $\mathcal{A}$ wins, i.e., $\mathsf{Sig.Verify}^H(\mathsf{vk}, m^*, \sigma^*) = \top$ and $\mathcal{A}$ have never queried m^* to the signing query.

Our goal is to prove $\Pr[\mathsf{T}_1] = \mathsf{negl}(\lambda)$.

Exp_2**:** In this experiment, the challenger picks a subset $S \subseteq \mathcal{M}$ by putting each $m \in \mathcal{M}$ into S with probability $\epsilon := \frac{1}{2q_{\mathsf{sig}}}$ (independently for each $m \in \mathcal{M}$). Let m_i be $\mathcal{A}$'s i-th signing query. As soon as the challenger detects an event $m_i \in S$ or $m^* \notin S$, it immediately aborts and returns 0. Except for this, this experiment is identical to the previous experiment.

For any fixed $m_1, ..., m_{q_{\mathsf{sig}}}, m^*$ such that $m^* \notin \{m_1, ..., m_{q_{\mathsf{sig}}}\}$, the probability that the challenger does not abort is at least $(1-\epsilon)^{q_{\mathsf{sig}}} \cdot \epsilon \geq \frac{\epsilon}{2} = \frac{1}{\mathsf{poly}(\lambda)}$. Therefore, we have $\Pr[\mathsf{T}_2] \geq \frac{1}{\mathsf{poly}(\lambda)} \Pr[\mathsf{T}_1]$.

Exp_3**:** In this experiment, the challenger samples H in a different way. Specifically, the challenger chooses $H_0, H_1 \xleftarrow{\$} \mathsf{Func}(\mathcal{X}, \mathcal{Y})$ and sets

$$H(x) := \begin{cases} H_0(x) & \text{if } \mathsf{XtoM}(x) \in S \\ H_1(x) & \text{otherwise} \end{cases}$$

where S is a subset sampled as in the previous experiment.

Since the distribution of H sampled as above is uniform over $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ for each fixed subset S, this experiment is identical to the previous one from $\mathcal{A}$'s view. Therefore, we have $\Pr[\mathsf{T}_3] = \Pr[\mathsf{T}_2]$.

Exp_4**:** In this experiment, the challenger samples H_1 in a different way. Specifically, the challenger chooses $\tilde{H} \xleftarrow{\$} \mathsf{Func}(\mathcal{X}, \mathcal{R})$ and sets $H_1(x) := F_{\mathsf{vk}}(\tilde{H}(x))$. The rest of the experiment is identical to the previous one.

By condition 3 of Theorem 4.11, the distribution of $H_1(x)$ in this game is statistically indistinguishable from the uniform distribution over $\mathcal{Y}$ and independent for each x. Therefore, by Lemma 2.2, we have $|\Pr[\mathsf{T}_4] - \Pr[\mathsf{T}_3]| = \mathsf{negl}(\lambda)$.

Exp_5**:** In this experiment, the challenger responds to signing queries in a different way. Specifically, when $\mathcal{A}$ makes a signing query m, if $m \in S$, the experiment aborts and returns 0 similarly to the previous experiment. Otherwise, the challenger returns $\sigma \xleftarrow{\$} \mathcal{S}^{\widetilde{H}}(\mathsf{vk}, m)$. The rest of the experiment is identical to the previous one.

The difference from the previous experiment is that when responding to a signing query, σ is generated by $\mathcal{S}^{\widetilde{H}}(\mathsf{vk}, m)$ instead of $\mathsf{Sig.Sign}^{H}(\mathsf{sigk}, m)$ when $m \notin S$. By the definition of H, we have $H(x) = H_1(x) = F_{\mathsf{vk}}(\widetilde{H}(x))$ for all queries x made by $\mathsf{Sig.Sign}^{H}(\mathsf{sigk}, m)$ for $m \notin S$. Therefore, by condition 3 of Theorem 4.11, the joint distribution of the random oracle H and all signatures generated by the signing oracle in these two experiments are statistically close. Therefore, we have $|\Pr[\mathsf{T}_5] - \Pr[\mathsf{T}_4]| = \mathsf{negl}(\lambda)$.

Exp_6**:** In this experiment, the challenger samples H in a different way, and the other part of the experiment is modified accordingly. Looking ahead, this modification is made for making it possible for a reduction algorithm to efficiently simulate H. Let Q be the maximal number of random oracle queries made in the experiment, $\widetilde{\mathcal{H}}$ be a family of $2Q$-wise independent hash functions from $\mathcal{X}$ to $\mathcal{R}$, a, b be positive integers such that $\left|\frac{b}{a} - \epsilon\right| = \mathsf{negl}(\lambda)$, and $\mathcal{H}_S$ be a family of $2Q$-wise hash functions from $\mathcal{X}$ to $[b]$. Then the challenger chooses $H_0 \xleftarrow{\$} \mathsf{Func}(\mathcal{X}, \mathcal{Y})$, $\widetilde{H} \xleftarrow{\$} \widetilde{\mathcal{H}}$, $H_S \xleftarrow{\$} \mathcal{H}_S$, and sets

$$H(x) := \begin{cases} H_0(x) & \text{if } H_S(\mathsf{XtoM}(x)) \leq a \\ F(\widetilde{H}(x)) & \text{otherwise} \end{cases}.$$

Moreover, whenever the previous experiment checks $m \in S$, this experiment instead checks $H_S(m) \leq a$. The rest of the experiment is identical to the previous experiment.

By Lemma 2.1, output distribution of the experiment does not change if we replace $\widetilde{H}$ and H_T with random functions. After this replacement, the experiment is identical to the previous one except that ϵ is replaced with $\frac{a}{b}$. Since we have $\left|\frac{b}{a} - \epsilon\right| = \mathsf{negl}(\lambda)$, by Lemma 2.2, we obtain $|\Pr[\mathsf{T}_6] - \Pr[\mathsf{T}_5]| = \mathsf{negl}(\lambda)$.

What is left is to prove $\Pr[\mathsf{T}_6] = \mathsf{negl}(\lambda)$. To prove this, we construct an adversary $\mathcal{B}$ that breaks the EUF-NMA security in the QROM assuming that $\Pr[\mathsf{T}_6]$ is non-negligible. For avoiding confusion, we denote by $G : \mathcal{X} \to \mathcal{Y}$ the random oracle used in the EUF-NMA game which $\mathcal{B}$ plays. Then $\mathcal{B}$ is described as follows:

$\mathcal{B}^G(\mathsf{vk})$**:** Given a verification key vk, it samples H similarly to Exp_6 except that it embeds G into H_0. More precisely, it samples $\widetilde{H}$ and H_S as in Exp_6 and sets

$$H(x) := \begin{cases} G(x) & \text{if } H_S(\mathsf{XtoM}(x)) \leq a \\ F(\widetilde{H}(x)) & \text{otherwise} \end{cases}.$$

Then it runs $\mathcal{A}$ giving an input vk and quantum access to H. (Note that quantum access to H can be efficiently simulated by quantum access to G along with $\widetilde{H}$ and H_S.) When $\mathcal{A}$ makes a signing query m, $\mathcal{B}$ aborts if $H_S(m) \leq a$, and otherwise returns $\sigma \xleftarrow{\$} \mathcal{S}^{\widetilde{H}}(\mathsf{vk}, m)$. Let (m^*, σ^*) be $\mathcal{A}$'s final output. $\mathcal{B}$ aborts if $H_S(m^*) > a$, and otherwise outputs (m^*, σ^*) as its final output.

We can see that $\mathcal{B}$ perfectly simulates the environment of Exp_6 for $\mathcal{A}$, and $\mathcal{B}$'s output (m^*, σ^*) satisfies $\mathsf{Sig.Verify}^H(\mathsf{vk}, m^*, \sigma^*) = \top$ if the experiment returns 1. Moreover, when the experiment returns 1, we have $H_S(m^*) > a$, which implies that all queries x made by $\mathsf{Sig.Verify}^H(\mathsf{vk}, m^*, \sigma^*)$ satisfies $H(x) = G(x)$ by condition 2 of Theorem 4.11 and the definition of H. Therefore, we have $\mathsf{Sig.Verify}^G(\mathsf{vk}, m^*, \sigma^*) = \top$. Therefore, $\mathcal{B}$ succeeds in winning the EUF-NMA game with probability $\Pr[\mathsf{T}_6]$. On the other hand, as noted at the beginning of this proof, the scheme is EUF-NMA secure against quantum adversaries in the QROM, and thus we have $\Pr[\mathsf{T}_6] = \mathsf{negl}(\lambda)$.

Combining the above, we have $\Pr[\mathsf{T}_1] = \mathsf{negl}(\lambda)$, which completes the proof of Theorem 4.11. □

4.5 Application to Quantum Query Lower Bounds

We use Theorem 4.3 to give a general theorem on quantum query lower bounds. Specifically, we prove the following theorem.

Theorem 4.12. *Let $\mathcal{X}$ and $\mathcal{Y}$ be sets, $H : \mathcal{X} \to \mathcal{Y}$ be a random function, k be a positive integer, and $R \subseteq \mathcal{Y}^k$ be a relation over $\mathcal{Y}^k$. Then for any q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr_H[(H(x_1), ..., H(x_k)) \in R \wedge x_j \neq x_{j'} \text{ for } j \neq j' : (x_1, ..., x_k) \xleftarrow{\$} \mathcal{A}^{|H\rangle}]$$
$$\leq (2q+1)^{2k} \Pr[\exists \pi \in \mathsf{Perm}([k]) \text{ s.t. } (y_{\pi(1)}, ..., y_{\pi(k)}) \in R : (y_1, ..., y_k) \xleftarrow{\$} \mathcal{Y}^k]$$

where $\mathsf{Perm}([k])$ *denotes the set of all permutations over* $[k]$.

Proof. We consider a (non-interactive) public-query search-type game where an adversary is given quantum access to a random oracle H and sends $(x_1, ..., x_k) \in \mathcal{X}^k$ to the challenger and the challenger outputs $\top$ if and only if $(H(x_1), ..., H(x_k)) \in R$ and $(x_1, ..., x_k)$ is pair-wise distinct. The LHS of the inequality in Theorem 4.12 is the probability that $\mathcal{A}$ wins the game. By Theorem 4.3, there exists a k-classical-query adversary $\mathcal{B}$ that wins the game while making exactly the same queries as those made by the challenger with probability at least $\frac{1}{(2q+1)^{2k}}$ times the probability that $\mathcal{A}$ wins. We observe that $\mathcal{B}$ makes exactly the same queries as the challenger if and only if it just sends a permutation of its k queries as the message $(x_1, ..., x_k)$. In this case, $\mathcal{B}$'s winning probability is at most $\Pr[\exists \pi \in \mathsf{Perm}([k]) \text{ s.t. } (y_{\pi(1)}, ..., y_{\pi(k)}) \in R : (y_1, ..., y_k) \xleftarrow{\$} \mathcal{Y}^k]$ since the random oracle values are uniformly and independently random over $\mathcal{Y}$. By combining the above, we obtain Theorem 4.12. □

In the following, we discuss applications of Theorem 4.12. In the following, we consider a random oracle $H : \{0,1\}^m \to \{0,1\}^n$.

Preimage Search. We first consider a preimage search problem, where an algorithm is given a quantum access to H and its goal is to find $x \in \{0,1\}^n$ such that $H(x) = 0^n$.[18] For a preimage search problem, we obtain the following lower bound.

Corollary 4.13 (Preimage Search)**.** *For any q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr_H[H(x) = 0^n : x \xleftarrow{\$} \mathcal{A}^{|H\rangle}] \leq \frac{(2q+1)^2}{2^n}.$$

In particular, for finding preimages of y_j for all $j \in [k]$ with a constant probability, $\mathcal{A}$ must make $\Omega(2^{n/2})$ queries.

Proof. Let $R \in \{0,1\}^n$ be a relation defined by $R := \{0^n\}$. Then we have $\Pr[y \in R : y \xleftarrow{\$} \{0,1\}^n] = 2^{-n}$. Then Theorem 4.12 implies the above corollary. □

This bound is asymptotically tight as it matches the upper bound by the Grover's algorithm. A similar lower bound in the worst case was proven in [BBBV97], and their proof can be easily extended to give a lower bound in the average case to prove the above corollary.

Multi-Preimage Search. Moreover, the above proof can be extended to give a lower bound for the multi-instance version of the preimage search problem. Specifically, we have the following corollary.

Corollary 4.14 (Multi-Preimage Search)**.** *For any $(y_1^*, ..., y_k^*) \in \{\{0,1\}^n\}^k$ such that $y_j^* \neq y_{j'}^*$ and any q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr_H[H(x_j) = y_j^* \text{ for all } j \in [k] : (x_1, ..., x_k) \xleftarrow{\$} \mathcal{A}^{|H\rangle}(y_1^*, ..., y_k^*)] \leq \frac{k!(2q+1)^{2k}}{2^{kn}}.$$

[18] 0^n can be any fixed string.

Proof. Let $R \in \{\{0,1\}^n\}^k$ be a relation defined by $R := \{(y_1^*, ..., y_k^*)\}$. Then for each fixed permutation $\pi \in \mathsf{Perm}([k])$, we have $\Pr[(y_{\pi(1)}, ..., y_{\pi(k)}) \in R : (y_1, ..., y_k) \xleftarrow{\$} \{\{0,1\}^n\}^k] = 2^{-kn}$. Since the number of permutations over $[k]$ is $k!$, Theorem 4.12 implies the above corollary. □

Intuitively, this means that the probability to find preimages of y_j for all $j \in [k]$ decreases exponentially in k in the range of $kq^2 = o(2^n)$.

Collision Finding. Next, we consider the collision-finding problem, which is a problem to find $x \neq x'$ such that $H(x) = H(x')$.

Corollary 4.15 (Collision-Finding)**.** *For any q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr_H[H(x) = H(x') \ \wedge \ x \neq x' : (x, x') \xleftarrow{\$} \mathcal{A}^{|H\rangle}] \leq \frac{(2q+1)^4}{2^n}.$$

In particular, for finding a collision with a constant probability, $\mathcal{A}$ must make $\Omega(2^{n/4})$ queries.

Proof. Let $R \in \{\{0,1\}^n\}^2$ be a relation defined by $R := \{(y, y)\}_{y \in \{0,1\}^n}$. Then we have $\Pr[(y_1, y_2) \in R : (y_1, y_2) \xleftarrow{\$} \{\{0,1\}^n\}^2] = 2^{-n}$. Noting that R is permutation-invariant, Theorem 4.12 implies the above corollary. □

This bound is non-tight since there is a collision-finding algorithm using $O(2^{n/3})$ [BHT98]. A tight lower bound was given in [AS04, Amb05] in the worst case and in [Zha15] in the average case.

Multi-Collision Finding. The above proof can be extended to give a lower bound for multi-collision finding.

Corollary 4.16 (Multi-Collision-Finding)**.** *For $k \in \mathbb{N}$ and $\mathbf{x} := (x_1, ..., x_k) \in \{\{0,1\}^n\}^k$, we say that $\mathbf{x}$ is a k-collision of H if we have $H(x_1) = ... = H(x_k)$ and $x_j \neq x_{j'}$ for all $j \neq j'$. For any q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr_H[\mathbf{x} \text{ is } k\text{-collision of } H : \mathbf{x} \xleftarrow{\$} \mathcal{A}^{|H\rangle}] \leq \frac{(2q+1)^{2k}}{2^{(k-1)n}}.$$

In particular, for finding a k-collision with a constant probability, $\mathcal{A}$ must make $\Omega\left(2^{\frac{(k-1)n}{2k}}\right)$ queries.

Proof. Let $R \in \{\{0,1\}^n\}^k$ be a relation defined by $R := \{(y, ..., y)\}_{y \in \{0,1\}^n}$. Then we have $\Pr[(y_1, ..., y_k) \in R : (y_1, ..., y_k) \xleftarrow{\$} \{\{0,1\}^n\}^k] = 2^{-(k-1)n}$. Noting that R is permutation-invariant, Theorem 4.12 implies the above corollary. □

This bound is non-tight since there is a k-collision-finding algorithm using $O\left(2^{\frac{(2^{k-1}-1)n}{2^k-1}}\right)$ queries [HSTX19, LZ19a]. A tight lower bound was given in

[LZ19a]. Though our lower bound is non-tight, it is non-trivial, and especially it approaches to $2^{n/2}$ when k becomes larger similarly to the tight bound. The proof of the tight lower bound given in [LZ19a] is rather complicated whereas our proof given above is extremely simple given Theorem 4.3.

Generalized Collision Finding. Our proof can be further extended to give a lower bound for the following more general problem.

Corollary 4.17 (Generalized Collision Finding)**.** *For $(k_1, ..., k_\ell) \in \mathbb{N}^\ell$ and $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_\ell) = ((x_{1,1}, ..., x_{1,k_1}), ..., (x_{\ell,1}, ..., x_{1,k_\ell})) \in \{\{0,1\}^m\}^{k_1} \times ... \times \{\{0,1\}^m\}^{k_\ell}$, we say that $\mathbf{x}$ is $(k_1, ..., k_\ell)$-collision of H if $\mathbf{x}_i$ is k_i-collision of H for all $i \in [\ell]$ and $x_{i,j} \neq x_{i',j'}$ for all $(i,j) \neq (i',j')$. For any q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr_H[\mathbf{x} \text{ is } (k_1, ..., k_\ell)\text{-collision of } H : \mathbf{x} \xleftarrow{\$} \mathcal{A}^{|H\rangle}] \leq \frac{k! \cdot (2q+1)^{2k}}{k_1!...k_\ell! \cdot 2^{(k-\ell)n}}$$

where $k := k_1 + ... + k_\ell$. In particular, when k is constant, for finding a $(k_1, ..., k_\ell)$-collision with a constant probability, $\mathcal{A}$ must make $\Omega\left(2^{\frac{(k-\ell)n}{2k}}\right)$ queries.

Proof. Let $R \in \{\{0,1\}^n\}^k$ be a relation consisting of all $\mathbf{y} = (y_1, ..., y_k)$ such that there exists a disjoint partition $\{S_j\}_{j\in[\ell]}$ of $[k]$ such that $|S_j| = k_j$ and for all $j, j' \in S_j$ we have $y_j = y_{j'}$. We can see that $\mathbf{x}$ is $(k_1, ..., k_\ell)$-collision if and only if $H(\mathbf{x}) \in R$ where $H(\mathbf{x})$ is a string obtained by applying H for each component of $\mathbf{x}$. By a simple combinatorial argument, we can see that the number of elements in R can be upper bounded by $\frac{k!}{k_1!...k_\ell!}2^{\ell n}$.[19] Then we have $\Pr[(y_1, ..., y_k) \in R : (y_1, ..., y_k) \xleftarrow{\$} \{\{0,1\}^n\}^k] = \frac{k!}{k_1!...k_\ell! \cdot 2^{(k-\ell)n}}$. Noting that R is permutation-invariant, Theorem 4.12 implies the above corollary. □

This bound is apparently non-tight. For example, in the case of $k_1 = ... = k_\ell = 1$ and $\ell = k$, the above corollary only gives a trivial bound. Nonetheless, we believe that this corollary also includes many non-trivial cases (e.g., multi-collision problem). It may be possible to give a better bound using more sophisticated techniques such as the compressed oracle technique [Zha19], but that would result in a complicated proof.

References

[AA14] Scott Aaronson and Andris Ambainis. The need for structure in quantum speedups. *Theory Comput.*, 10:133–166, 2014.

[Aar10] Scott Aaronson. BQP and the polynomial hierarchy. In Leonard J. Schulman, editor, *42nd ACM STOC*, pages 141–150. ACM Press, June 2010.

[19] This is not the exact number since some strings (e.g., k-collisions) are counted many times.

[AGKZ20] Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 255–268. ACM Press, June 2020.

[Agr19] Shweta Agrawal. Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 191–225. Springer, Heidelberg, May 2019.

[Amb05] Andris Ambainis. Polynomial degree and lower bounds in quantum complexity: Collision and element distinctness with small range. *Theory Comput.*, 1(1):37–46, 2005.

[AP20] Shweta Agrawal and Alice Pellet-Mary. Indistinguishability obfuscation without maps: Attacks and fixes for noisy linear FE. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 110–140. Springer, Heidelberg, May 2020.

[ARU14] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *55th FOCS*, pages 474–483. IEEE Computer Society Press, October 2014.

[AS04] Scott Aaronson and Yaoyun Shi. Quantum lower bounds for the collision and the element distinctness problems. *J. ACM*, 51(4):595–605, 2004.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh V. Vazirani. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 26(5):1510–1523, 1997.

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4):778–797, 2001.

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th FOCS*, pages 320–331. IEEE Computer Society Press, October 2018.

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

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. *IACR Cryptol. ePrint Arch.*, 2020:1024, 2020.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In Claudio L. Lucchesi and Arnaldo V. Moura, editors, *LATIN '98: Theoretical Informatics, Third Latin American Symposium, Campinas, Brazil, April, 20-24, 1998, Proceedings*, volume 1380 of *Lecture Notes in Computer Science*, pages 163–169. Springer, 1998.

[BKVV20] Zvika Brakerski, Venkata Koppula, Umesh V. Vazirani, and Thomas Vidick. Simpler proofs of quantumness. In *TQC 2020*, volume 158 of *LIPIcs*, pages 8:1–8:14, 2020.

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

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111. Springer, Heidelberg, May 1995.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.

[BZ13] Dan Boneh and Mark Zhandry. Secure signatures and chosen ciphertext security in a quantum computing world. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 361–379. Springer, Heidelberg, August 2013.

[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.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1):80–101, January 2013.

[FOPS01] Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA-OAEP is secure under the RSA assumption. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 260–274. Springer, Heidelberg, August 2001.

[FR99] Lance Fortnow and John D. Rogers. Complexity limitations on quantum computation. *J. Comput. Syst. Sci.*, 59(2):240–252, 1999.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

[GP20] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. Cryptology ePrint Archive, Report 2020/1010, 2020. https://eprint.iacr.org/2020/1010.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[HSTX19] Akinori Hosoyamada, Yu Sasaki, Seiichiro Tani, and Keita Xagawa. Improved quantum multicollision-finding algorithm. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*, pages 350–367. Springer, Heidelberg, 2019.

[JZC+18] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 96–125. Springer, Heidelberg, August 2018.

[KLS18] Eike Kiltz, Vadim Lyubashevsky, and Christian Schaffner. A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 552–586. Springer, Heidelberg, April / May 2018.

[KS20] Juliane Krämer and Patrick Struck. Encryption schemes using random oracles: From classical to post-quantum security. In Jintai Ding and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*, pages 539–558. Springer, Heidelberg, 2020.

[KYY18] Shuichi Katsumata, Shota Yamada, and Takashi Yamakawa. Tighter security proofs for GPV-IBE in the quantum random oracle model. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 253–282. Springer, Heidelberg, December 2018.

[LZ19a] Qipeng Liu and Mark Zhandry. On finding quantum multi-collisions. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 189–218. Springer, Heidelberg, May 2019.

[LZ19b] Qipeng Liu and Mark Zhandry. Revisiting post-quantum Fiat-Shamir. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 326–355. Springer, Heidelberg, August 2019.

[Mah18] Urmila Mahadev. Classical homomorphic encryption for quantum circuits. In Mikkel Thorup, editor, *59th FOCS*, pages 332–338. IEEE Computer Society Press, October 2018.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, Heidelberg, May 1996.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.

[Son14] Fang Song. A note on quantum security for post-quantum cryptography. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014*, pages 246–265. Springer, Heidelberg, October 2014.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 192–216. Springer, Heidelberg, October / November 2016.

[WW20] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious lwe sampling. *IACR Cryptol. ePrint Arch.*, 2020:1042, 2020.

[YZ20] Takashi Yamakawa and Mark Zhandry. A note on separating classical and quantum random oracles. Cryptology ePrint Archive, Report 2020/787, 2020. https://eprint.iacr.org/2020/787.

[Zha12] 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.

[Zha15] Mark Zhandry. A note on the quantum collision and set equality problems. *Quantum Inf. Comput.*, 15(7&8):557–567, 2015.

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

[ZYF+19] Jiang Zhang, Yu Yu, Dengguo Feng, Shuqin Fan, and Zhenfeng Zhang. On the (quantum) random oracle methodology: New separations and more. Cryptology ePrint Archive, Report 2019/1101, 2019. https://eprint.iacr.org/2019/1101.

A Construction of ECRH for General Predicates

In this section, we prove Lemma 3.6, that is, we construct a family of ECRH for general predicates relative to a classical oracle.

First, we prepare several notations and a lemma taken from [AGKZ20].

Definition A.1. (Affine Partition Function [AGKZ20, Definition 2]) *Let n be a positive even integer and $P = (A_y)_{y\in\{0,1\}^{n/2}}$ be an $n/2$-ordered affine partition of $\mathbb{F}_2^n$. That is, A_y is an affine subspace of $\mathbb{F}_2^n$ of dimension $n/2$ for each $y \in \{0,1\}^{n/2}$ and they are pairwise disjoint. We denote by $A_y^\perp$ the orthogonal complement of the linear subspace corresponding to A_y. We define an affine partition function $(G_P, G_P^\perp)$ as follows:*[20]

1. *$G_P : \mathbb{F}_2^n \to \{0,1\}^{n/2}$ such that $G_P(x) = y$ if and only if $x \in A_y$.*
2. *$G_P^\perp : \mathbb{F}_2^n \times \{0,1\}^{n/2} \to \{0,1\}$ such that $G_P^\perp(x,y) = 1$ if and only if $x \in A_y^\perp$.*

Lemma A.2. (Collision Resistance [AGKZ20, Theorem 4]) *There exists an $n/2$-ordered affine partition P of $\mathbb{F}_2^n$ such that for any $o(2^{n/4})$-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr[G_P(x) = G_P(x') : (x,x') \xleftarrow{\$} \mathcal{A}^{|G_P\rangle,|G_P^\perp\rangle}] \leq \mathsf{negl}(n).$$

By slightly modifying the proof of equivocality in [AGKZ20, Theorem 5], we prove the following lemma.

Lemma A.3 (Equivocality)**.** *Let $P = (A_y)_{y\in\{0,1\}^{n/2}}$ be an $n/2$-ordered affine partition of $\mathbb{F}_2^n$ and $t \in \mathbb{N}$ be a positive integer. Then, there exists an oracle-aided*

[20] We use G instead of H unlike [AGKZ20] for avoiding a confusion with a random oracle.

quantum algorithm $\mathcal{E}_t$ *with running time* $\mathsf{poly}(\lambda, t)$ *such that for any* $b \in \{0,1\}$ *and* $y \in \{0,1\}^{n/2}$*, if we have*

$$\Pr_{x \xleftarrow{\$} A_y}[p(x) = b] \geq t^{-1},$$

then we have

$$\Pr[x \in A_y \wedge p(x) = b : x \xleftarrow{\$} \mathcal{E}_t^{|G_P^\perp(\cdot,y)\rangle, |p\rangle}(|A_y\rangle, b)] = 1 - \mathsf{negl}(n)$$

where $|A_y\rangle$ *is the uniform superposition over all elements of* A_y*.*

Proof. The main idea is to run the Grover's algorithm by implementing the reflection over $|A_y\rangle$ by using a quantum oracle access to $G_P^\perp(\cdot, y)$ similarly to [AGKZ20]. For implementing the Grover's algorithm, we need the following two unitaries:

- $O_b = 2\sum_{x \in \mathbb{F}_2^n : p(x) = b} |x\rangle\langle x| - I$ and
- $U_y = 2|A_y\rangle\langle A_y| - I = F(2|A_y^\perp\rangle\langle A_y^\perp| - I)F$ where F is the quantum Fourier transform over $\mathbb{F}_2^n$ (which is equivalent to the Hadamard transform).

We can easily implement O_b by using a quantum oracle access to p. On the other hand, we cannot impelement U_y using quantum access to the oracle $G_P^\perp(\cdot, y)$. However, as implicitly shown in [AGKZ20], we can implement an operator that works similarly to U_y in the subspace spanned by $\{|x\rangle\}_{x \in A_y}$. Since the state remains in the subspace spanned by $\{|x\rangle\}_{x \in A_y}$ throughout the execution of the Grover's algorithm, we can use that operator instead of U_y. By our assumption, the (squared) amplitude of $|A_y\rangle$ on "correct answers" (i.e., $x \in A_y$ such that $p(x) = b$) is at least t, the Grover's algorithm finds a correct answer within $O(\sqrt{t})$ iterations with overwhelming probability. We note that this can be done with a single copy of $|A_y\rangle$ even though (even an approximation of) the number of correct answers is unknown to $\mathcal{E}_t$ since we can reuse the state as shown in [ARU14, Appendix E.1].

Thus, what is left is to implement an operator that works similarly to U_y on any state in the space spanned by $\{|x\rangle\}_{x \in A_y}$. For this, we prove the following claim.

Claim A.4. *For a state* $|\psi\rangle = \sum_{x \in A_y} \alpha_x |x\rangle$*, we have*

$$FU_{A_y^\perp}F|\psi\rangle = U_y|\psi\rangle.$$

where $U_{A_y^\perp}$ *is a unitary such that*

$$U_{A_y^\perp}|x\rangle = \begin{cases} |x\rangle & \text{If } x \in A_y^\perp \\ -|x\rangle & \text{otherwise} \end{cases}.$$

We note that $U_{A_y^\perp}$ *can be implemented by a single quantum access to* $G_P^\perp(\cdot, y)$*.*

Proof. (of Claim A.4) It suffices to prove that we have

$$FU_{A_y^\perp}F\left|A_y\right\rangle = \left|A_y\right\rangle \tag{9}$$

and for any $\left|\psi\right\rangle = \sum_{x\in A_y}\alpha_x\left|x\right\rangle$ that is orthogonal to $\left|A_y\right\rangle$, we have

$$FU_{A_y^\perp}F\left|\psi\right\rangle = -\left|\psi\right\rangle. \tag{10}$$

Since A_y is an affine subspace, there is a corresponding linear subspace S_y and a translation $t\in\mathbb{F}_2^n$ such that we have $A_y = S_y + t$. By using this, we can show Eq. 9 by the following calculation:

$$\begin{aligned}
FU_{A_y^\perp}F\left|A_y\right\rangle &= FU_{A_y^\perp}\frac{1}{2^{n/4}}\sum_{z\in A_y^\perp}(-1)^{t\cdot z}\left|z\right\rangle\\
&= F\frac{1}{2^{n/4}}\sum_{z\in A_y^\perp}(-1)^{t\cdot z}\left|z\right\rangle\\
&= \left|A_y\right\rangle.
\end{aligned}$$

On the other hand, for any $\left|\psi\right\rangle = \sum_{x\in A_y}\alpha_x\left|x\right\rangle$ that is orthogonal to $\left|A_y\right\rangle$, we have

$$F\left|\psi\right\rangle = \frac{1}{2^{n/2}}\sum_{z\in\mathbb{F}_2^n}\sum_{x\in A_y}\alpha_x(-1)^{x\cdot z}\left|z\right\rangle.$$

For any $z\in A_y^\perp$, since we have $x\cdot z = t\cdot z$ for all $x\in A_y$, the amplitude of $F\left|\psi\right\rangle$ on z is

$$\frac{1}{2^{n/2}}\sum_{x\in A_y}\alpha_x(-1)^{x\cdot z} = \frac{1}{2^{n/2}}\sum_{x\in A_y}\alpha_x(-1)^{t\cdot z} = 0$$

since we have $\sum_{x\in A_y}\alpha_x = 0$ by the orthogonality between $\left|\psi\right\rangle$ and $\left|A_y\right\rangle$. Therefore, $U_{A_y^\perp}$ just gives a phase of (-1) to $F\left|\psi\right\rangle$, and thus we have Eq. 10. □

This completes the proof of Lemma A.3. □

Above lemmas immediately give a construction of ECRH for general predicates relative to G_P and $G_P^\perp$ as follows where we set $n := 2\lambda$:

$\mathsf{ECRH.Setup}^{G_P,G_P^\perp}(1^\lambda)$**:** It outputs $\mathsf{crs} := 1^\lambda$.[21]

$\mathsf{ECRH.Eval}^{G_P,G_P^\perp}(\mathsf{crs},x)$**:** It outputs $y := G_P(x)$.

[21] If we consider a heuristic instantiation in the standard model based on obfuscation, it outputs obfuscation of oracles G_P and $G_P^\perp$ as a common reference string.

$\mathsf{ECRH.Gen}^{|G_P\rangle,|G_P^\perp\rangle}(\mathsf{crs})$**:** It generates a uniform superposition over $\mathbb{F}_2^n$, evaluates G_P in superposition to generate a state

$$\sum_{x\in\mathbb{F}_2^n} |x\rangle\,|G_P(x)\rangle\,,$$

measures the second register to obtain an outcome $y \in \{0,1\}^{n/2}$ along with the collapsed state $|A_y\rangle = \sum_{x\in A_y} |x\rangle$, and outputs y and $|\mathsf{sk}\rangle := |A_y\rangle$.

$\mathsf{ECRH.Equiv}^{|G_P\rangle,|G_P^\perp\rangle,|p\rangle}(1^t, |\mathsf{sk}\rangle, b)$**:** It runs $x \xleftarrow{\$} \mathcal{E}_t^{|G_P^\perp\rangle,|p\rangle}(|\mathsf{sk}\rangle = |A_y\rangle, b)$ and outputs x.

Correctness follows from Lemma A.3. By Lemma A.2, there exists a choice of P such that the above ECRH satisfies the collision resistance. Moreover, we can see that $\{x \in \mathbb{F}_2^n : \mathsf{ECRH.Eval}(\mathsf{crs}, x) = y\} = A_y$ and thus its size is $2^{n/2} = 2^\lambda$. This completes the proof of Lemma 3.6

B Proof for Soundness of Publicly Verifiable PoQRO

In this section, we give a full proof of Lemma 3.7. That is, we give a proof for the soundness of the publicly verifiable PoQRO in Sec. 3.1.2.

For proving soundness of the PoQRO, we consider the following 4-round protocol between a prover with quantum access to a random oracle $H : \{0,1\}^{2\lambda} \to \{0,1\}$ and a classical verifier with classical access to H as an intermediate tool.

First Message: The verifier generates $\mathsf{crs} \xleftarrow{\$} \mathsf{ECRH.Setup}(1^\lambda)$ and sends crs to the prover as a first message.

Second Message: The prover generates $(y_i, |\mathsf{sk}_i\rangle) \xleftarrow{\$} \mathsf{ECRH.Gen}(\mathsf{crs})$ for all $i \in [m]$, sends $\{y_i\}_{i\in[m]}$ to the verifier as a second message and keeps $\{|\mathsf{sk}_i\rangle\}_{i\in[m]}$ as its internal state.

Third Message: The verifier randomly picks $c_i \xleftarrow{\$} \{0,1\}$ for all $i \in [m]$ and sends $\{c_i\}_{i\in[m]}$ to the prover as a third message.

Fourth Message: The prover generates $x_i \xleftarrow{\$} \mathsf{ECRH.Equiv}^{|H\rangle}(1^3, |\mathsf{sk}_i\rangle, c_i)$ for all $i \in [m]$ and sends $\{x_i\}_{i\in[m]}$ to the verifier as a fourth message.

Verification: The verifier accepts if we have $\mathsf{ECRH.Eval}(\mathsf{crs}, x_i) = y_i$ and $H(x_i) = c_i$ for all $i \in [m]$. Otherwise, it rejects.

In the following, we call the above protocol $\Pi_{\mathsf{4\text{-}round}}$ and the PoQRO protocol in Sec. 3.1.2 Π_{PoQRO}. For an adversary $\mathcal{A}$ against these protocols, we say that $\mathcal{A}$ wins if it succeeds in letting the verifier accept.

In the following, we prove the following lemmas:

Lemma B.1. *If there exists an efficient quantum cheating prover with classical access to random oracles H and H' in Π_{PoQRO} that wins with a non-negligible probability, then there exists an efficient quantum cheating prover with classical access to H in $\Pi_{4\text{-round}}$ that wins with a non-negligible probability. This holds relative to any oracles.*

Lemma B.2. *Any quantum efficient quantum cheating prover with classical access to H in $\Pi_{4\text{-round}}$ cannot wins with a non-negligible probability assuming the collision-resistance of the underlying ECRH. This holds relative to any oracles.*

By combining Lemma B.1 and B.2, Lemma 3.7 immediately follows. In the following, we prove Lemma B.1 and B.2.

Proof. (of Lemma B.1.) Let $\mathcal{A}$ be an efficient quantum cheating prover of Π_{PoQRO} that makes at most q classical queries to H' and wins with a non-negligible probability. Without loss of generality, we can assume that $\mathcal{A}$ queries $y_1||...||y_m$ to H' where $\{(x_i, y_i)\}_{i\in[m]}$ is the proof output by $\mathcal{A}$ by increasing its number of queries to H' to $q+1$. We also assume that $\mathcal{A}$ does not make the same query twice. Then we construct a cheating prover $\mathcal{B}$ with classical access to H in $\Pi_{4\text{-round}}$ that wins with a non-negligible probability as follows:

$\mathcal{B}^H$**:** This algorithm works as follows:

1. Uniformly choose $i^* \xleftarrow{\$} [q+1]$.
2. Given the first message crs from the verifier, it runs $\mathcal{A}$ on input crs until it makes i^*-th query to H' while simulating the random oracles to $\mathcal{A}$ as follows:
   (a) The oracle H is simulated by just forwarding queries to the external oracle H.
   (b) The oracle H' is simulated on the fly for the first i^*-1 queries. That is, whenever $\mathcal{A}$ makes a query to H', $\mathcal{B}$ uniformly chooses a string from $\{0,1\}^m$ and returns it as the response by H'.[22]

   Let $y_1^*||...||y_m^*$ be $\mathcal{A}$'s i^*-th query to H'.
3. Send $\{y_i^*\}_{i\in[m]}$ as the second message to the external verifier, and receive the third message $\{c_i\}_{i\in[m]}$ from the verifier.
4. Return $c := c_1||...||c_m$ to $\mathcal{A}$ as the response to the i^*-th query to H'.
5. Run the rest of the execution of $\mathcal{A}$ while simulating oracles similarly as in Step 2.
6. Let $\{(x_i, y_i)\}_{i\in[m]}$ be $\mathcal{A}$'s final output. If we have $\{y_i^*\}_{i\in[m]} \neq \{y_i\}_{i\in[m]}$, return $\perp$ and abort. Otherwise, send $\{(x_i)\}_{i\in[m]}$ as the fourth message to the external verifier.

[22] $\mathcal{B}$ need not record queries since we assume that $\mathcal{A}$ does not make the same query twice.

We can see that $\mathcal{B}$ perfectly simulates the environment of the soundness game of Π_{PoQRO} to $\mathcal{A}$. Since $\mathcal{B}$ sets $H'(y_1^*||...||y_m^*) := c$, if we have $\{y_i^*\}_{i\in[m]} = \{y_i\}_{i\in[m]}$, $\mathcal{B}$ wins whenever $\mathcal{A}$ wins. Moreover, since we assume that $\mathcal{A}$ queries $y_1||...||y_m$ to H', we have $\{y_i^*\}_{i\in[m]} = \{y_i\}_{i\in[m]}$ with probability $\frac{1}{q+1}$ over the choice of i^* fixing all other randomness. Therefore, the probability that $\mathcal{B}$ wins is equal to $\frac{1}{q+1}$ times the probability that $\mathcal{B}$ wins, which is non-negligible. Clearly, this reduction works relative to any oracle. This completes the proof of Lemma B.1. □

Proof. (of Lemma B.2.) Suppose that there exists an efficient quantum cheating prover $\mathcal{A}$ with classical access to H in $\Pi_{\mathsf{4\text{-}round}}$ that wins with a non-negligible probability. Since $\mathcal{A}$ only makes classical queries to H, we can run the execution between $\mathcal{A}$ and the verifier by recording all queries to H. We denote by L the list of queries by $\mathcal{A}$ to the oracle H. Let crs, $\{y_i\}_{i\in[m]}$, $\{c_i\}_{i\in[m]}$, and $\{x_i\}_{i\in[m]}$ be first, second, third, and fourth messages in the execution. Without loss of generality, we assume that $\mathcal{A}$ queries x_i to H for all $i \in [m]$. We denote by Col the event that L contains $x \neq x'$ such that $\mathsf{ECRH.Eval}(\mathsf{crs}, x) = \mathsf{ECRH.Eval}(\mathsf{crs}, x')$. By a straightforward reduction to the collision resistance of ECRH, we have $\Pr[\mathsf{Col}] = \mathsf{negl}(\lambda)$. Moreover, conditioned on that Col does not happen, L contains a unique preimage of y_i for all $i \in [m]$, which should be equal to x_i. In the following, we first prove that the probability that the verifier accepts is at most 2^{-m} conditioned on that Col does not occur and $\{y_i\}_{i\in[m]}$ is pairwise distinct, and after that we explain how to prove the same proeprty without assuming the latter. Suppose that Col does not occur and $\{y_i\}_{i\in[m]}$ is pairwise distinct. For each $i \in [m]$, we say that x_i is determined when one of the following happens:

1. When $\mathcal{A}$ sends the second message $\{y_i\}_{i\in[m]}$, the list L contains x_i' such that $\mathsf{ECRH.Eval}(\mathsf{crs}, x_i') = y_i$,
2. When $\mathcal{A}$ makes a random oracle query x_i' after receiving the third message, we have $\mathsf{ECRH.Eval}(\mathsf{crs}, x_i') = y_i$.

Conditioned on that Col does not happen, for each $i \in [m]$, x_i is determined exactly once (by either of the above cases), and we have $x_i = x_i'$ where x_i' is as above. Let $I \subseteq [m]$ be a subset consisting of all i such that x_i is determined by the first case. For each $i \in [I]$, since c_i is randomly chosen after x_i is determined (and thus $H(x_i)$ is determined), we have $\Pr[H(x_i) = c_i] = 1/2$ (over the randomness of c_i). For each $i \in [m] \setminus I$, when x_i is determined, c_i is already fixed. However, for this case, we can rely on the uniformity of random oracle values to obtain $\Pr[H(x_i) = c_i] = 1/2$ (over the randomness of $H(x_i)$). Moreover, since we assume that $\{y_i\}_{i\in[m]}$ is pairwise distinct, $\{x_i\}_{i\in[m]}$ is pairwise distinct and thus the event that $H(x_i) = c_i$ is independent for each $i \in [m]$. Therefore, the probability that $H(x_i) = c_i$ holds for all $i \in [m]$ is 2^{-m}.

When $\{y_i\}_{i\in[m]}$ is not pairwise distinct, though the first case works similarly, the second case does not work as it is since $\{x_i\}_{i\in[m]}$ may not be pairwise distinct and thus the event $H(x_i) = c_i$ is not independent for each $i \in [m]$. However, the proof can be modified by the following observation: For any y^*,

let $S_{y^*} \subseteq [k]$ be the set of i such that $y_i = y^*$. Then, conditioned on that Col does not occur, we must have $x_i = x^*$ for all $i \in S_{y^*}$ for some x^*. Therefore, for satisfying $H(x_i) = c_i$ for all $i \in S_{y^*}$ (which is equivalent to $H(x^*) = c_i$ for all $i \in S_{y^*}$), we must have $c_i = c^*$ for all $i \in S_{y^*}$ for some c^*, which happens with probability $2^{-|S_{y^*}|+1}$. Moreover, for any fixed $\{c_i\}_{i \in S_{y^*}}$ that satisfies the above, the probability that we have $H(x^*) = c^*$ is $1/2$ over the randomness of $H(x^*)$. Combining the above, the probability that we have $H(x_i) = c_i$ for all $i \in S_{y^*}$ is $2^{-|S_{y^*}|}$. Moreover this event is independent for different y^*. Based on this observation, we can conclude that conditioned on that Col does not occur, the probability that $H(x_i) = c_i$ holds for all $i \in [m]$ is 2^{-m}.

Therefore, the overall probability that verifier accepts is $2^{-m} + \mathsf{negl}(\lambda) = \mathsf{negl}(\lambda)$.

Finally, we remark that the above reduction works relative to any oracle since the reduction just uses $\mathcal{A}$ in a black-box manner. □

C Applications of Theorem 4.11

In this section, we give examples of digital signature schemes that satisfy the conditions of Theorem 4.11. We only show that these schemes actually satisfies the conditions of Theorem 4.11, and omit security proofs themselves since they are already known to be secure in the QROM. The purpose of this section is to show the applicability of Theorem 4.11.

C.1 FDH Signatures.

Let $H : \mathcal{X} \to \mathcal{Y}$ be the random oracle and $\mathcal{F}$ be a family of trapdoor permutations over $\mathcal{Y}$. Then the FDH signature scheme over a message space $\mathcal{X}$ is described as follows:

$\mathsf{Sig.KeyGen}(1^\lambda)$**:** This algorithm generates a permutation f from $\mathcal{F}$ along with its trapdoor td. Then it outputs $\mathsf{vk} := f$ and $\mathsf{sigk} := \mathsf{td}$.

$\mathsf{Sig.Sign}^H(\mathsf{sigk} = \mathsf{td}, m)$**:** This algorithm computes $\sigma := f^{-1}(H(m))$ by using td and outputs it.

$\mathsf{Sig.Verify}^H(\mathsf{vk} = f, m, \sigma)$**:** This algorithm returns $\top$ if and only if $f(\sigma) = H(m)$ holds.

We can see that this scheme satisfies the conditions of Theorem 4.11 as follows:

1. Clearly, $\mathsf{Sig.KeyGen}$ does not make a random oracle query and $\mathsf{Sig.Verify}$ makes 1 random oracle query.
2. Since random oracle queries by $\mathsf{Sig.Sign}$ and $\mathsf{Sig.Verify}$ are the message itself, we can just define XtoM as the identity function.

3. This condition is satisfied by defining $\mathcal{R} := \mathcal{Y}$, $F := f$, and $\mathcal{S}^{\widetilde{H}}(\mathsf{vk}, m)$ as an algorithm that outputs $\widetilde{m}$.

The lattice-based variant of the FDH signatures called the GPV signatures [GPV08] has a similar syntax and thus satisfies the conditions of Theorem 4.11 as well.

C.2 Fiat-Shamir Signatures

As a building block, we use an identification protocol (which is derived from a Σ-protocol) with the following syntax.

- A setup algorithm Setup on input 1^λ generates a public key pk and sk, and gives sk to the prover and pk to the verifier.
- The prover first stage algorithm P_1 is given a secret key sk generates a "commitment" com along with its opening open. We denote this procedure by $(\mathsf{com}, \mathsf{open}) \xleftarrow{\$} P_1(\mathsf{sk})$. Then it sends $\mathsf{com} \in \mathcal{COM}$ to the verifier.
- The verifier chooses a "challenge" $\mathsf{chal} \xleftarrow{\$} \mathcal{CHAL}$ and sends chal to the prover.
- The prover generates an "answer" ans. We denote this procedure by $\mathsf{ans} \xleftarrow{\$} P_2(\mathsf{open}, \mathsf{chal})$.
- The verifier (deterministically) outputs $\top$ or $\bot$. We denote this procedure by $V(\mathsf{pk}, \mathsf{com}, \mathsf{chal}, \mathsf{ans})$.

Honest Verifier Zero-Knowledge. We require the identification scheme to satisfy the honest-verifier zero-knowledge that requires the following: There exists a classical efficient algorithm $\mathcal{S}_{\mathsf{ID}}$ such that for an overwhelming fraction of $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{Setup}(1^\lambda)$, we have

$$\left\{ (\mathsf{com}, \mathsf{chal}, \mathsf{ans}) : \begin{array}{l} (\mathsf{com}, \mathsf{open}) \xleftarrow{\$} P_1(\mathsf{sk}), \\ \mathsf{chal} \xleftarrow{\$} \mathcal{CHAL} \\ \mathsf{ans} \xleftarrow{\$} P_2(\mathsf{open}, \mathsf{chal}) \end{array} \right\}$$
$$\approx \left\{ (\mathsf{com}, \mathsf{chal}, \mathsf{ans}) : (\mathsf{com}, \mathsf{chal}, \mathsf{ans}) \xleftarrow{\$} \mathcal{S}_{\mathsf{ID}}(\mathsf{pk}) \right\}$$

where $\approx$ means that two distributions are statistically indistinguishable.

High Min-Entropy.[23] We require an identification protocol to satisfy the following property: For an overwhelming fraction of $(\mathsf{pk}, \mathsf{sk})$ generated by $\mathsf{Setup}(1^\lambda)$, for any com^*, we have

$$\Pr[\mathsf{com} = \mathsf{com}^* : (\mathsf{com}, \mathsf{open}) \xleftarrow{\$} P_1(\mathsf{sk})] = \mathsf{negl}(\lambda).$$

[23] Though this property is assumed in some existing works on Fiat-Shamir signatures (e.g., [KLS18]), this is sometimes not a default requirement for an identification scheme for applying Fiat-Shamir transform. We discuss how to remove this requirement at the end of this section.

Soundness Usually, we also need to require that soundness which roughly requires that a cheating prover who does not know sk cannot let the verifier accept with non-negligible probability. On the other hand, since the purpose of this section is just to show that the Fiat-Shamir signatures satisfies the conditions of Theorem 4.11, we do not need soundness. Therefore, we omit the formal definition of soundness.

Let $H : \mathcal{M} \times \mathcal{COM} \to \mathcal{CHAL}$ be a random oracle where $\mathcal{M}$ is the message space of the Fiat-Shamir signatures. Then the Fiat-Shamir signature scheme is described as follows:

$\mathsf{Sig.KeyGen}(1^\lambda)$**:** This algorithm generates $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$ and outputs $\mathsf{vk} := \mathsf{pk}$ and $\mathsf{sigk} := \mathsf{sk}$.

$\mathsf{Sig.Sign}^H(\mathsf{sigk} = \mathsf{sk}, m)$**:** This algorithm computes $(\mathsf{com}, \mathsf{open}) \xleftarrow{\$} P_1(\mathsf{sk})$, $\mathsf{chal} := H(m||\mathsf{com})$, $\mathsf{ans} \xleftarrow{\$} P_2(\mathsf{open}, \mathsf{chal})$, and outputs $\sigma := (\mathsf{com}, \mathsf{ans})$.

$\mathsf{Sig.Verify}^H(\mathsf{vk} = \mathsf{pk}, m, \sigma = (\mathsf{com}, \mathsf{ans}))$**:** This algorithm computes $\mathsf{chal} := H(m||\mathsf{com})$ and returns $\top$ if and only if $V(\mathsf{pk}, \mathsf{com}, \mathsf{chal}, \mathsf{ans}) = \top$.

We can see that this scheme satisfies the conditions of Theorem 4.11 as follows:

1. Clearly, $\mathsf{Sig.KeyGen}$ does not make a random oracle query and $\mathsf{Sig.Verify}$ makes only 1 random oracle query.
2. The condition is clearly satisfied if we define $\mathsf{XtoM}(m||\mathsf{com}) := m$.
3. We let $\mathcal{R}$ be the randomness space of $\mathcal{S}_{\mathsf{ID}}$, and define F_{vk} and $\mathcal{S}$ as follows:

   $F_{\mathsf{vk}}(r)$**:** It computes $(\mathsf{com}, \mathsf{chal}, \mathsf{ans}) := \mathcal{S}_{\mathsf{ID}}(\mathsf{vk}; r)$ and outputs chal.

   $\mathcal{S}^{\tilde{H}}(\mathsf{vk} = \mathsf{pk}, m)$**:** It computes $(\mathsf{com}, \mathsf{chal}, \mathsf{ans}) := \mathcal{S}_{\mathsf{ID}}(\mathsf{vk}; r)$ and outputs $\sigma := (\mathsf{com}, \mathsf{ans})$.

   For any $(m_1, ..., m_\ell)$ (which is not necessarily pair-wise distinct), when generating signatures for these messages, the same random query is not made twice with an overwhelming probability due to the the high min-entropy property. Therefore, we can prove the desired property by a standard hybrid argument using the honest-verifier zero-knowledge property.

On Removing the High Min-Entropy Property. The high min-entropy property is used to ensure that the same random query is not made twice when generating signatures for $m_1, ..., m_\ell$ with overwhelming probability. Since signing procedures for different messages do not make the same random oracle queries, if we assume that $(m_1, ..., m_\ell)$ is pair-wise distinct, then the high min-entropy property is not needed. This can be assumed without loss of generality if we consider a derandomized version of the above scheme where signing algorithm derives its randomness by PRF on input m. Thus, the derandomized version of the Fiat-Shamir signatures satisfies the conditions of Theorem 4.11 without assuming the high min-entropy property for the underlying identification scheme.