On the (Quantum) Random Oracle Methodology: New Separations and More

Jiang Zhang[1], Yu Yu[2], Dengguo Feng[1], Shuqin Fan[1], and Zhenfeng Zhang[3]

[1] State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University
[3] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences, China
{jiangzhang09@gmail.com, yuyu@yuyu.hk, feng@tca.iscas.ac.cn, shuqinfan78@163.com,
zfzhang@tca.iscas.ac.cn

Abstract. In this paper, we first give, to the best of our knowledge, the first exponential separation between the ROM and QROM. Technically, we will first present a simple and efficient quantum distinguisher $\mathcal{D}_q$ which can recognize the QROM by making at most two quantum RO queries, and can only be cheated by an adversary making (sub-)exponential classical RO queries. This (sub-)exponential query gap allows us to remove the "unit time" and "zero time" assumptions that are crucially needed for previously known separation due to Boneh et al. (Asiacrypt 2011). The construction of our distinguisher relies on a new *information versus disturbance* lemma, which may be of independent interest. Moreover, we show that the quantum operations of $\mathcal{D}_q$ can actually be delegated to any quantum algorithms in a way that can be efficiently verified by a classical verifier under the LWE assumption, which allows us to give a pure classical distinguisher $\mathcal{D}_c$ that can efficiently distinguish an environment equipped with a RO from that with a QRO. By using $\mathcal{D}_c$ as a black-box, we can transform schemes that are secure in the ROM but insecure in the QROM (under the LWE assumption).

We further abstract a class of BB-reductions in the ROM under the notion of committed-programming reduction (CPRed) for which the simulation of the RO can be easily quantized to handle quantum queries (from the adversary in the QROM). We show that 1) some well-known schemes such as the FDH signature and the Boneh-Franklin identity-based encryption are provably secure under CPReds; and 2) a CPRed associated with an instance-extraction algorithm implies a reduction in the QROM, which subsumes several recent results such as the security of the FDH signature by Zhandry (Crypto 2012) and the KEM variants from the Fujisaki-Okamoto transform by Jiang et al. (Crypto 2018).

We finally show that CPReds are incomparable to non-programming reductions (NPReds) and randomly-programming reductions (RPReds) formalized by Fischlin et al. (Asiacrypt 2010), which gives new insights into the abilities (e.g., observability and programmability) provided by the (Q)ROM, and the hardness of proving security in the QROM.

1 Introduction

In the random oracle model (ROM), all parties, including the adversary, are given access to an "idealized" random function (i.e., a random oracle). The ROM has been widely used to analyze the security of many (well-known) cryptosystems, but as pointed out by Boneh et al. [7], the classical ROM is problematic when considering quantum adversaries. This actually motivates them to introduce the quantum ROM (QROM) [7], where honest parties (e.g., the cryptosystems in post-quantum cryptography) typically use the random oracle (RO) in a classical way, but the adversary is explicitly allowed to send quantum queries to the RO. Boneh et al. [7] justified the necessity of the QROM by presenting an artificial identification protocol which is secure in the ROM but is insecure in the QROM. We observe that their results [7] raise several interesting problems.

First, the identification protocol for separating the QROM from the ROM in [7] is designed to directly utilize the gap in finding a collision of an n-bit output hash function between using the birthday attack with complexity $O(2^{n/2})$ in a classical way and using the Grover algorithm with complexity $O(2^{n/3})$ in a quantum way [19,11], which makes their arguments very sensitive to a "precise" timing model. In particular, they have to make sure that the running time of the artificial identification protocol is longer than $O(2^{n/3})$ "unit time"

for a quantum adversary to run the Grover algorithm [19], but is shorter than $O(2^{n/2})$ "unit time" for a classical adversary to implement the birthday attack. *One might wonder if there are other separations between the QROM and the ROM.*

Second, it is well-known that a cryptosystem secure in the ROM might not be secure in the standard model [12,26,25]. Combining this with Boneh et al.'s separation [7] that there is a scheme which is secure in the ROM but is insecure in the QROM, *it is natural to ask what the relation between the QROM and the standard model is, or whether the security of a cryptosystem in the QROM could somehow guarantee the security of the cryptosystem in the real world.*

Third, following the introduction of the QROM [7], several papers [31,28,22] have been devoted to giving new security reductions in the QROM for some well-known schemes that were already proven secure in the ROM. However, most of them [7,31,28,22] rely on ad hoc techniques, which are usually specific on the concrete design of the schemes, rather than the inherent properties of existing reductions in the ROM.[4] *Is there a class of (black-box) reductions in the ROM, which can be amended to handle quantum adversaries in the QROM? If yes, what is the relation between this class and those known in the literature?*

1.1 Our Results

Motivated by the fact that in the QROM [7], honest parties (e.g., the cryptosytems) typically use the random oracle (RO) in a classical way while the adversary can make quantum queries to the RO, we reformalize the classical RO (CRO) and the quantum RO (QRO) by equipping a RO with two interfaces: a *private* one for the honest parties, and a (quantum) *public* one for the adversary. Specifically, we think of a CRO $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot))$ as being a primitive with two classical and equal interfaces (i.e., $\mathcal{O}^c_{priv}(\cdot) = \mathcal{O}^c_{pub}(\cdot)$), and a QRO $\mathcal{O}^q = (O^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot))$ as being a primitive where the private interface $O^q_{priv}(\cdot)$ is classical but the public interface $\mathcal{O}^q_{pub}(\cdot)$ accepts quantum queries from the adversary. Moreover, a scheme in the ROM (QROM) is treated as a (quantum) polynomial time construction which uses a CRO (QRO) as a component.

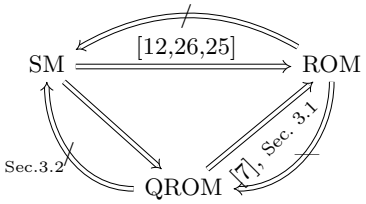

Fig. 1. Relations among the ROM, the QROM and the standard model (SM): $A \Rightarrow B$ (resp., $A \not\Rightarrow B$) means that the security in A always implies (resp., does not necessarily imply) that in B.

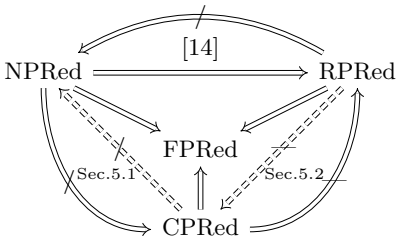

Fig. 2. Relations among the NPRed, the RPRed, the FPRed and the CPRed: $X \Rightarrow Y$ (resp., $X \not\Rightarrow Y$) means that the existence of X implies (resp., does not necessarily imply) that of Y.

We then adapt the indifferentiability framework of Maurer et al. [25] to the quantum setting, and separate the QROM from the ROM by showing that QRO is differentiable from CRO. Technically, we construct an efficient quantum algorithm $\mathcal{D}_q$ such that for all (even unbounded) algorithm $\mathcal{P}$ making at most a polynomial number of queries to a CRO $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot))$, $\mathcal{D}_q$ can distinguish a real QRO $\mathcal{O}^q = (O^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot))$ from the simulated one $\mathcal{O}' = (\mathcal{O}^c_{priv}(\cdot), \mathcal{P}^{\mathcal{O}^c_{pub}(\cdot)}(\cdot))$. In other words, no QPT adversary $\mathcal{P}$ in a CRO environment can cheat $\mathcal{D}_q$ that it is in a QRO environment by simulating a quantum interface for the CRO. Moreover, one can easily construct an (artificial) system such that it is secure in the ROM but is insecure in the QROM, by using $\mathcal{D}_q$ as a subroutine, and letting the system execute a malicious action (e.g., leaking the secret) if and only if $\mathcal{D}_q$ detects that the whole system is running in a QRO environment. For a full relation among the ROM, the QROM and the standard model (see Fig. 1), we also separate the

[4] We note that the history-free reduction in [7] was introduced to abstract a class of existing BB-reductions restricted to signatures in the ROM.

standard model from the QROM by adapting the separation between the standard model and the ROM in [25]. Then, we show that the quantum operations of $\mathcal{D}_q$ can be delegated to any quantum algorithms in a way that can be efficiently verified by a classical verifier, which allows us to obtain a classical distinguisher $\mathcal{D}_c$ between ROM and QROM. Using $\mathcal{D}_c$ as a sub-routine, one can construct artificial schemes that are secure in the ROM but insecure in the QROM without the "computational and timing assumptions", by simply letting the schemes do some malicious behaviors (e.g., leaking the secret key) if $\mathcal{D}_c$ thinks it is in the QROM, otherwise behave securely.

We further abstract a class of black-box (BB) reductions in the ROM under the notion of committed-programming reduction (CPRed) for which the simulation of the RO can be easily quantized. We show that 1) some well-known schemes [4,18,30,20] such as the full-domain hash (FDH) signature [4] and the Boneh-Franklin identity-based encryption [8] are provable under CPReds, and 2) a CPRed associated with an instance-extraction algorithm implies a reduction in the QROM, which not only subsumes several recent results [7,35,22] such as the security of the FDH from trapdoor permutations (TDP) by Zhandry [35] and the "implicit-rejection" KEM variants from the Fujisaki-Okamoto transform by Jiang et al. [22], but also gives new security reductions for other schemes such as the "implicit-rejection" KEM variants from TDPs [30] in the QROM.

We finally show that CPReds are incomparable to both non-programming reductions (NPReds) and randomly-programming reductions (RPReds) formalized by Fischlin et al. [14] (see Fig. 2), which gives new insights into the abilities (e.g., observability and programmability) provided by the (Q)ROM, and the hardness of proving security in the QROM (e.g., a scheme which is provably secure under NPReds[5] might still not be provable in the QROM as a reduction in the QROM seems to be limited in both observability and programmability). Technically, we show that the OAEP encryption from TDPs [3] which was proven secure under NPReds by Fujisaki et al. [17] is not provable under CPReds, and that the FDH signature [4] which was shown not to be provable under RPReds by Fischlin et al. [14] is provably secure under CPReds. Since a NPRed always implies a RPRed, we obtain a full relation in Fig. 2, where full-programming reductions (FPReds) denote all BB-reductions in the ROM.

1.2 Technical Overview

QRO is differentiable from CRO. We first observe that in the (Q)ROM the adversary typically can access the RO either directly by sending a (quantum) RO query or indirectly by sending a crypto-oracle query (e.g., a signing query) which may internally cause RO queries. Moreover, the internal RO queries are usually classical for most (even post-quantum) cryptosystems, and it is common that a signature/ciphertext/session-key contains the responses to some internal RO queries (e.g., [13,23,31]). This fact motivates us to formalize the RO by using the indifferentiability framework of Maurer et al. [25], which equips a primitive with two interfaces: a *private* one for honest parties (e.g., the cryptosystems) and a *public* one for the adversary. Informally, this framework aims at capturing the fact that an adversary against a cryptosystem may also have a (public) interface to access a primitive which is internally used by the cryptosytem. Take the Merkle-Damgård hash construction H^h (e.g., MD5, SHA1 and SHA2) which internally uses a publicly available compression function h as an example, in addition to having access to H^h (which is modeled by the private interface), an adversary against H^h also has public access to h (which is modeled by the public interface) as usually in practice.

Informally, let $\mathcal{F}_{n,m} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ be a family of functions for some positive integers n and m, we think of a classical RO (CRO) $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$ as being a primitive with two equal interfaces for both honest parties and adversaries satisfying $\mathcal{O}^c_{priv}(x) = \mathcal{O}^c_{pub}(x)$ for all $x \in \{0,1\}^n$, and a quantum RO (QRO) $\mathcal{O}^q = (O^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$ as being a primitive with a classical private interface $O^q_{priv}(\cdot)$ for honest parties and a consistent quantum public interface $\mathcal{O}^q_{pub}(\cdot)$ for adversaries satisfying that for a quantum query $|x, 0_m\rangle$ with arbitrary $x \in \{0,1\}^n$ to $\mathcal{O}^q_{pub}(\cdot)$, the measurement outcome on the response

[5] Note that a NPRed can observe the RO queries and responses between the adversary and an external RO (accessible by all parties), but cannot program the RO responses.

$\mathcal{O}^q_{pub}(|x,0_m\rangle)$ is always equal to a consistent classical pair $(x, O^q_{priv}(x))$. Moreover, a scheme in the ROM (QROM) is treated as a (quantum) polynomial time construction which uses a CRO (QRO) as a component.

In order to separate the QROM from ROM, by extending the result in [25] it suffices to construct a quantum polynomial time (QPT) algorithm $\mathcal{D}_q$ such that for all QPT algorithm $\mathcal{P}$ given access to (the public interface of) a CRO $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot))$, $\mathcal{D}$ can distinguish a QRO $\mathcal{O}^q = (O^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot))$ from a simulated one $\mathcal{O}' = (\mathcal{O}^c_{priv}(\cdot), \mathcal{P}^{\mathcal{O}^c_{pub}(\cdot)}(\cdot))$. Since there is a (polynomial) query gap for (quantum) algorithms to invert an n-bit input function (resp., find a collision of an m-bit output hash function) between using $O(2^n)$ (resp., $O(2^{m/2})$) classical queries and using $O(2^{n/2})$ (resp., $O(2^{m/3})$) quantum queries [19,11], one can build a trivial distinguisher $\mathcal{D}_q$. In fact, the first (and possibly only known) separation due to Boneh et al. [7] basically uses such a distinguisher, which requires extra "computational and timing assumptions" to utilize this polynomial query gap. Instead, we construct an efficient distinguisher $\mathcal{D}_q$ making at most two quantum queries and having exponential query gap.

Our starting point is that $\mathcal{D}_q$ can encode exponentially many classical queries into a single quantum query $|\phi\rangle = \sum_{x\in\{0,1\}^n} |x,0_m\rangle$ in superposition, while $\mathcal{P}^{\mathcal{O}^c_{pub}(\cdot)}$ can only make a polynomial number of classical queries to $\mathcal{O}^c_{pub}(\cdot)$. In particular, if D sends a quantum query $|\phi\rangle$ to $\mathcal{P}$, it is infeasible for $\mathcal{P}$ to compute a "valid" response $|\psi\rangle = \sum_{x\in\{0,1\}^n} \left|x, \mathcal{O}^c_{pub}(x)\right\rangle = \sum_{x\in\{0,1\}^n} |x, \mathcal{O}^c_{priv}(x)\rangle$. One main problem is that $\mathcal{D}$ cannot check if a response from $\mathcal{P}^{\mathcal{O}^c_{pub}(\cdot)}$ is valid or not, as by the quantum uncertainty principle it cannot extract all the classical pairs $\{(x, \mathcal{O}^c_{priv}(x))\}_{x\in\{0,1\}^n}$ from the quantum state $|\psi\rangle$.

To get around the above obstacle, we let $\mathcal{D}_q$ send a query $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$ using a random subset $X \subseteq \{0,1\}^n$, which ensures that a measurement on any "valid" response $|\psi\rangle = \sum_{x\in X} \left|x, \mathcal{O}^c_{pub}(x)\right\rangle = \sum_{x\in X} |x, \mathcal{O}^c_{priv}(x)\rangle$ to $|\phi\rangle$ always results in a pair $(x, \mathcal{O}^c_{priv}(x))$ satisfying $x \in X$. Clearly, if we could show that $\mathcal{P}$ cannot obtain sufficient information of X from $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$ to make a query $x \in X$ to $\mathcal{O}^c_{pub}(\cdot)$, then the proof is finished. However, this is not achievable as $\mathcal{P}$ can easily obtain an element $x \in X$ by simply measuring the query state $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$. Fortunately, we can show that it is infeasible for $\mathcal{P}$ to obtain sufficient information of X without significantly disturbing the query state $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$. Technically, we will establish a new *information versus disturbance* lemma (see Lemma 3), which gives a quantitative connection between the information about X that an algorithm obtains from a random quantum state $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$ and the disturbance that it causes to the state $|\phi\rangle$. Informally, the lemma says that for certain choice of X, any (even unbounded) algorithm given a state $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$ can only determine an element $x \in X$ by disturbing the state $|\phi\rangle$ in a way that can be efficiently detected. The proof of this lemma needs some techniques adapted from the security proof [6] of the BB84 quantum key distribution [5].

Here comes our distinguisher $\mathcal{D}_q^{\tilde{\mathcal{O}}_{priv}(\cdot),\tilde{\mathcal{O}}_{pub}(\cdot)}$: first sample a random state $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$ as in the above lemma, and send it to $\tilde{\mathcal{O}}_{pub}(\cdot)$; upon receiving the response $|\psi\rangle$, flip a random bit to either check that a measurement on $|\psi\rangle$ always results in a consistent pair $(x, \tilde{\mathcal{O}}_{priv}(x))$ for some $x \in X$ (which needs a classical query x to $\tilde{\mathcal{O}}_{priv}(\cdot)$), or check that the query state $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$ is not disturbed (which needs a quantum query to $\tilde{\mathcal{O}}_{pub}(\cdot)$ to "uncompute" $|\psi\rangle$) as in the above lemma; output 0 if either check fails, otherwise output 1.

Note that if $\mathcal{D}^{\tilde{\mathcal{O}}_{priv}(\cdot),\tilde{\mathcal{O}}_{pub}(\cdot)}$ is given access to a QRO $(\tilde{\mathcal{O}}_{priv}(\cdot), \tilde{\mathcal{O}}_{pub}(\cdot)) = (O^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot))$, it will always output 1 (since a QRO will always answer correctly without disturbing the query state). However, if it is given a simulated one $(\tilde{\mathcal{O}}_{priv}(\cdot), \tilde{\mathcal{O}}_{pub}(\cdot)) = (\mathcal{O}^c_{priv}(\cdot), \mathcal{P}^{\mathcal{O}^c_{pub}(\cdot)}(\cdot))$, a successful $\mathcal{P}^{\mathcal{O}^c_{pub}(\cdot)}$ needs to make at least one query with some $x \in X$ (to pass the first check) while not disturbing the query state $|\phi\rangle = \sum_{x\in X} |x,0_m\rangle$ in a detectable way (to pass the second check). As $\mathcal{P}^{\mathcal{O}^c_{pub}(\cdot)}$ is only allowed to make a polynomial number of queries[6], a successful $\mathcal{P}$ must determine an $x \in X$ without significantly disturbing the query state, which is infeasible by our *information versus disturbance* lemma. Thus, one of the checks must fail with noticeable probability for any algorithm $\mathcal{P}$ making a polynomial number of oracle queries.

[6] This requirement is crucial, because an algorithm making (sub-)exponential number of queries can always send a query with some $x \in X$ by brute-force search.

In order to give a classical distinguisher between ROM and QROM, our basic idea is to delegate the quantum computation of $\mathcal{D}_q$ to any quantum algorithm $\mathcal{P}$. At first glance, one might think it is trivial as Mahadev [24] showed that classical verification of any efficient quantum computations can be achieved under the LWE assumptions. However, this intuition does not work because the results in [24] uses an assumption that the quantum computation can be implemented by a public quantum circuits consisting polynomial number of simple universal quantum gates, while in our case 1) $\mathcal{D}_q$ is a oracle quantum algorithm and can only access the RO in a black-box way (and thus cannot be represented by quantum circuits with simple quantum gates); 2) the soundness of $\mathcal{D}_q$ relies on the fact that $\mathcal{D}_q$ has some secret information that cannot known by $\mathcal{P}$ (and thus cannot be encoded into a public quantum circuits).

Our solution is to divide the computation of $\mathcal{D}_q$ into small ones that do not exist the above two problems, and use some new techniques to glue up the small ones such that the whole protocol is still verifiable. Technically, we will show that there exists a protocol between a classical algorithm and a quantum algorithm to generate a quantum query with a random pastern that was only known by the classical algorithm by applying the quantum randomness generation protocol in [10]. Second, we give a protocol which can check if $\mathcal{P}$ makes a real QRO query based on an new observation of the measurement protocol in [24].

Formalizations of CPReds. Many classical security proofs in the ROM relies on adaptively programming the RO responses, which usually requires security reductions to copy and store the RO queries from the adversary. However, it is impossible to copy and store the adversary's quantum RO queries due to the quantum no-cloning theorem. As observed by Zhandry [**?**], this difficulty has led the literatures to use reductions which basic fix (the simulation strategy of) the RO at the very beginning. To capture this common feature, we motivate and introduce the notion of committed-programming reduction (CPRed) to abstract a class of BB-reductions in the ROM for which the RO simulation strategy is fixed before interacting with the adversary and can somehow be isolated from other behaviors of the reduction (such that we can focus on the RO simulation strategy and check the influence on the reduction if the adversary is allowed to make quantum RO queries).

A common reason of using ROs is that the reduction cannot normally answer some crypto-oracle queries (e.g., the FDH signing queries) due to the embedding of the target problem (e.g., inverting a TDP instance). Existing reductions in the literature usually apply the following two RO simulation strategies to successfully answer the crypto-oracle queries: 1) the reduction simulates the RO in a particular way such that the adversary may distinguish the simulated RO from a real one, but the crypto-oracle queries can be successfully answered w.r.t. the simulated RO if certain conditions on the adversary's queries are satisfied; or 2) the reduction perfectly simulates the RO for the adversary, but it will deviate from the simulated RO in answering some crypto-oracle queries by (implicitly) replacing the responses of the RO at some unknown points (to which it cannot directly compute the RO responses) with random values, which means that the adversary can detect this inconsistency by making a RO query with any one of those points (and in this case the reduction usually expects to obtain one of the unknown points by interacting with the adversary to solve its own problem).

We now restrict our attention to the reductions that only apply the above two strategies to simulate ROs. As there may exist many ROs, we allow the reduction to apply different strategies to different ROs (but only apply one of the two strategies to each RO). We distinguish the ROs for a given reduction $\mathcal{S}$ into two types, and say that a RO is a Type-I RO if $\mathcal{S}$ applies the first simulation strategy to it, otherwise a Type-II RO if $\mathcal{S}$ applies the second one to it. Then, we define a splittable property, and say that a reduction $\mathcal{S}$ in the ROM is a splittable reduction if it can be split into four sub-algorithms $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ (see Fig. 6): an initialization sub-algorithm $\mathcal{S}_1$ which takes as input a challenge instance of the underlying hard problem and produces some inputs to $\mathcal{S}_2$ and $\mathcal{S}_3$; a sub-algorithm $\mathcal{S}_2$ which takes inputs from $\mathcal{S}_1$ and handles all the crypto-oracle queries (e.g., the signing queries) from the adversary (and maybe abort if certain conditions on the adversary's queries are not satisfied for the simulation of Type-I RO); a sub-algorithm $\mathcal{S}_3$ which takes some inputs from $\mathcal{S}_1$ and handles the RO queries from the adversary (and may randomly output one of the Type-II RO to enable the reduction to obtain some unknown points); and a finalization algorithm $\mathcal{S}_4$ which produces a solution to the challenge instance by using the outputs from $\mathcal{S}_2$ and $\mathcal{S}_3$. Intuitively, $\mathcal{S}_1$ and $\mathcal{S}_4$ are two "internal" sub-algorithms for the initial and final work of $\mathcal{S}$, while $\mathcal{S}_2$ and $\mathcal{S}_3$ are two "external"

sub-algorithms for directly interacting with adversaries and handling the queries from the adversaries (note that we allow both $\mathcal{S}_2$ and $\mathcal{S}_3$ to abort or terminate before $\mathcal{A}$ terminates). Clearly, $\mathcal{S}_3$ represents all the RO simulation strategy of $\mathcal{S}$, and $\mathcal{S}$ can only make use of the adversary's ability via the outputs of $\mathcal{S}_2$ and $\mathcal{S}_3$ to solve its own challenge. The isolation of the RO simulation from other behaviors is achieved by not allowing $\mathcal{S}_2$ and $\mathcal{S}_3$ to interact with each other.

To formally capture the feature that the RO simulation strategies are fixed before interacting with the adversary, we associate a reduction with a deterministic function $F(\cdot,\cdot)$ for Type-I RO and a fixed index function $I(\cdot)$ for Type-II RO. Specifically, we say that $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ is (F, I)-splittable reduction (see Definition 7) if: 1) $\mathcal{S}_3$ will only use the deterministic function $F(\tau, \cdot)$ with some input τ from $\mathcal{S}_1$ to answer Type-I RO queries, and a real random function $f(\cdot)$ from $\mathcal{S}_1$ for Type-II RO queries (i.e., $\mathcal{S}_3$ will either use $F(\tau, \cdot)$ or $f(\cdot)$ to answer a RO query); and 2) given a crypto-oracle query z, $\mathcal{S}_2$ will only deviate at some Type-II RO queries whose positions are specified by $I(z)$ from the Type-II RO simulated by $\mathcal{S}_3$ (i.e., $I(z)$ specifies the positions of all the Type-II RO queries that deviate from the simulated Type-II RO in answering a crypto-oracle query z) by replacing the responses to those RO queries with random values.

A committed-programming reduction (CPRed) is basically a parameterized (F_λ, I)-splittable reduction for $\lambda \in (0, 1)$, satisfying some necessary conditions (see Definition 8) such as the probability that $\mathcal{S}_2$ will not abort is noticeable (to ensure the non-triviality of the reduction), and the view of the any adversary given access to the simulated Type-I RO using $F_\lambda(\tau, \cdot)$ is not far from the case that it is given a real RO (otherwise the adversary can distinguish the simulated game from the real one by simply making Type I RO queries). The term "committed-programming" comes from the facts that $F_\lambda(\cdot,\cdot)$ and $I(\cdot)$ are deterministic and fixed before interacting with the adversary, and that the reduction does not adaptively program the RO for $\mathcal{S}_2$ to handle the crypto-oracle queries.

To demonstrate the non-triviality and usefulness of CPReds, we will show that some well-known schemes [4,8,18,30,20] such as the FDH signature [4] and the Boneh-Franklin IBE [8] are provably secure under CPReds.

Implications of CPReds. Now, consider a CPRed $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ from problem P to problem Q, namely, $\mathcal{S}$ can solve problem P by using a black-box adversary solving problem Q. The intuition behind that the CPRed $\mathcal{S}$ can be lifted to a reduction from P to Q in the QROM is simple: as the RO simulation (i.e., $\mathcal{S}_3$) of $\mathcal{S}$ is independent from other behaviors, and the simulation of crypto-oracles (i.e., $\mathcal{S}_2$) is oblivious to the RO queries from the adversary $\mathcal{A}$, it seems harmless when $\mathcal{A}$ is given quantum access to the (simulated) RO; moreover, because $\mathcal{S}_3$ is essentially implemented by using functions $F(\tau, \cdot)$ and $f(\cdot)$, we can directly quantize both functions F and f to handle quantum RO queries from $\mathcal{A}$. Indeed, our goal is to show that the reduction $\mathcal{S}' = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}'_3, \mathcal{S}_4)$ is a valid quantum reduction from P to Q, where $\mathcal{S}'_3$ is a quantized version of $\mathcal{S}_3$.

In order to achieve the above goal, we require the CPRed $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ to have two extra properties (for simplicity, we will merge the two properties into a single one in Theorem 10): 1) the view of the quantum algorithm given access to the simulated Type-I RO using $F(\tau, \cdot)$ is not far from the case that it is given a real quantum RO (note that the Type II RO simulated using a random function $f(\cdot)$ is identical to a real RO); 2) $\mathcal{S}$ has an associated instance-extraction algorithm, which can extract an instance of P (together with some auxiliary information that is needed by $\mathcal{S}_2$ and $\mathcal{S}'_3$) from an instance of Q. This first property is natural and intuitive, as otherwise the adversary can distinguish the security game of Q simulated by $\mathcal{S}'$ from the real one by simply making quantum RO queries. The second property is somewhat artificial for proving that $\mathcal{S}' = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}'_3, \mathcal{S}_4)$ is a valid reduction from P to Q in the QROM, which needs us to establish a direct connection between the simulated game of Q by $\mathcal{S}'$ and the real one. However, in the real security game of Q there is essentially no conception of P instance, while in the simulated game of Q by $\mathcal{S}'$ a P instance is explicitly needed to run $\mathcal{S}_1$ and generate the inputs (e.g., τ) for $\mathcal{S}_2$ and $\mathcal{S}'_3$ (to compute $F(\tau, \cdot)$). To connect the above two games via game sequences in a formal proof, we need to embed a P instance into the security game of Q and generate the necessary information that is needed by $\mathcal{S}_2$ and $\mathcal{S}'_3$ in some intermediate game. The instance-extraction algorithm provides us a natural way to do it, and it exists for some well-known existing schemes. Take the FDH signature from TDP [4] as an example, given a challenge verification key vk^* of the FDH signature which basically is a trapdoor permutation function index, one can naturally extract a problem instance of inverting the underlying trapdoor permutation by outputting vk

together with a uniformly random point y^* chosen from the range of the permutation (and other information needed for $\mathcal{S}_2$ and $\mathcal{S}_3$ can be generated by running $\mathcal{S}_1$ with inputs (vk, y^*)). We note that the history-free reduction [7] directly contains a similar instance-extraction algorithm (restricted to the signatures) in the definition basically for the same reason.

With the above two properties, the proof that $\mathcal{S}' = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}'_3, \mathcal{S}_4)$ is a valid reduction from P to Q in the QROM is intuitive and direct except that the one-way to hiding (O2H) lemma [34,20,**?**] is carefully applied to handle the case that $\mathcal{S}_2$ may deviate from the Type-II RO simulation at the RO queries with positions specified by $I(z)$ in answering a crypto-oracle query z. Briefly, the O2H lemma says that if the adversary can make some quantum RO queries to detect the inconsistency between $\mathcal{S}_2$ and $\mathcal{S}_3$ at some Type-II RO query h with non-negligible probability, then one can recover h from the quantum RO queries of the adversary with non-negligible probability (which is sufficient for $\mathcal{S}'$ to solve its own problem). We note that our reduction $\mathcal{S}'$ not only subsumes the reductions for the FDH signature [4] and the GPV-IBE scheme in [35], for the PSF-FDH signature [18] in [7], and for the "implicit rejections" KEM variants from the FO transform [16,20] given in [22], but also gives new security reductions for the "implicit-rejection" KEM variants from TDP [30] in the QROM.

Incomparability of CPReds. We also study the relation between CPReds and the notions of ROM BB-reductions formalized by Fischlin et al. [14]. We show that 1) a NPRed does not imply a CPRed (i.e., NPRed $\not\Rightarrow$ CPRed); and 2) a CPRed does not imply a RPRed (i.e., CPRed $\not\Rightarrow$ RPRed). As a NPRed always implies a RPRed (i.e., NPRed $\Rightarrow$ RPRed) [14], we immediately obtain that CPRed $\not\Rightarrow$ NPRed, and RPRed $\not\Rightarrow$ CPRed. This means that CPReds are incomparable to both NPReds and RPReds. Technically, our results are achieved by investigating the security reductions of two well-known schemes, namely, the OAEP encryption from TDP [3] and the FDH signature [4].

First, we show that the OAEP encryption from TDP [3], which was provably secure under NPReds by Fujisaki et al. [17], is not provable under CPReds. The key point is that the reduction [17] for the CCA-security of the OAEP encryption crucially relies on the observability of the RO queries to answer the decryption queries, which is not achievable for CPReds (as in a CPRed the sub-algorithm for simulating the decryption oracle does not interact with the sub-algorithm for simulating the RO, and cannot use the RO queries from the adversary to answer the decryption queries). We will use the two-oracle separation technique of Hsiao and Reyzin [21] to formally rule out the existence of CPReds for the CCA-security of the OAEP encryption.

Second, we show that the FDH signature [4], which was shown not to be provable under RPReds by Fischlin et al. [14], is provably secure under CPReds. Our main observation is that existing reductions for the unforgeability of the FDH signature need to program the RO response such that the corresponding preimage under some permutation is known (in any signing query). This is not achievable for a RPRed since it cannot directly set the RO responses, but is achievable for a CPRed since the simulation of the RO can be done by using a sub-algorithm equipped with two random functions: one for guessing if a RO query from the adversary will be used in a signing query, and the other for picking a random preimage to program the RO response if the guess is "yes".

1.3 Related Work and Discussion

The random oracle methodology was first formalized by Bellare and Rogaway [2] and has been widely used to design and analyze many schemes such as the OAEP encryption [3] and the FDH signature [4]. Although most "honestly-designed" schemes in the ROM seem to keep the security in practice, but the soundness of this methodology has been questioned by the literatures [12,26,25]. The first separation between the ROM and the standard model was given by Canetti, Goldreich and Halevi [12], who showed that there exist signature and encryption schemes that are secure in the ROM, but for which any implementation of the RO results in insecure schemes. Later, Maurer et al. [25] introduced the notion of indifferentiability, and gave a more simple separation.

There are two main abilities that a black-box (BB) reduction can get from the ROM: the observability (i.e., the ability to see all the RO queries [1]) and the programmability (i.e., the ability to set the RO responses [14]).

In 2010, Fischlin et al. [14] first formalized three notions of BB-reductions with different programmability in the ROM: fully-programming reduction (FPRed), randomly-programming reduction (RPRed) and non-programming reduction (NPRed). Later, Ananth and Bhaskar [1] considered the security of several existing schemes under BB-reductions without observability in the ROM.

Boneh et al. [7] first introduced the quantum ROM (QROM), and separate the QROM from the ROM by giving an "artificial" identification protocol which is provably secure in the ROM, but is insecure in the QROM. Their separation crucially relies on the speedup of the Grover quantum algorithm [19,11] over classical algorithms in finding a collision of hash functions (and a precise timing model), while we resort to a technically different way and construct an efficient algorithm which can distinguish a CRO environment from a QRO environment by directly using the quantum mechanics. Boneh et al. [7] also considered a class of BB-reductions (which are restricted to signatures) in the ROM, i.e., history-free reductions, and show that a signature scheme with a history-free reduction may still imply the security in the QROM.

Following [7], many researchers have devoted to giving security proofs for existing schemes [35,13,31,20,23,22] and designing new schemes [32,33,28] in the QROM. However, as commented in [14,31], it might be hard to prove the security of the OAEP encryption [3] in the QROM. We note that our results essentially provide some new insights into this "hypothesis" as we have showed that the OAEP encryption cannot be proven secure under CPReds, and that a CPRed with an instance-extraction algorithm implies a reduction in the QROM. Note that Targhi and Unruh [31] proved the security of a variant of the OAEP encryption in the QROM by adding an additional hash (modeled as a RO) to each ciphertext, which basically can be seen as a technique to "force the adversary to submit the input" of the RO to the reduction and thus force the adversary to provide a direct connection between the RO queries and the decryption queries.

1.4 Open Problems and Future Work

One problem is to find new applications of CPReds and obtain security proofs for more schemes in the QROM. Another problem is to generalize our notion of CPReds to obtain a larger class of BB-reductions that subsume the results in [13,33,**?**] which seem to have no CPReds in the ROM but are already known to be provably secure in the QROM, and establish more connections between the proof techniques in the classical ROM and those in the QROM.

2 Preliminaries

Let κ be the security parameter. The standard notations O, ω are used to classify the growth of functions. Denote log as the logarithm with base 2. A function $f(\kappa)$ is negligible in κ if for every positive c, we have $f(\kappa) < \kappa^{-c}$ for sufficiently large κ. By $\mathrm{negl}(\kappa)$ we denote an arbitrary negligible function. The notation $\xleftarrow{\$}$ denotes randomly choosing elements from a distribution (or the uniform distribution over a finite set). Denote ϵ (resp., $\emptyset$) as an empty string (resp., set).

Let $\mathbb{C}$ be the set of complex numbers, and let $\mathbb{C}^N$ be the complex vector space of N dimension, where $N \geq 1$ is an integer. The bra-ket notations of $\langle\cdot|$ and $|\cdot\rangle$ are used to denote row and column vectors in $\mathbb{C}^N$, respectively. For any vector $v \in \mathbb{C}^N$, v^T denote the transpose of v, and v^* denotes the conjugate transpose of v. For any vectors $w = (w_0, \ldots, w_{N-1})^T, v = (v_0, \ldots, v_{N-1})^T \in \mathbb{C}^N$, the inner product between w and v is defined as $\langle w|v\rangle = \sum_{i=0}^{N-1} w_i^* v_i \in \mathbb{C}$.

2.1 Quantum Computation

We briefly recall some background for quantum computation, and refer to [27] for more information. Formally, a quantum system $\mathcal{Q}$ with N configurations labeled by $\{0, \ldots, N-1\}$ is associated to the Hilbert space $\mathcal{H}_N = \mathbb{C}^N$ with the inner product $\langle w|v\rangle = \sum_{i=0}^{N-1} w_i^* v_i \in \mathbb{C}$. A pure state of $\mathcal{Q}$ is specified by a column vector $|\phi\rangle \in \mathcal{H}_N$ of norm 1 (i.e., $\langle\phi|\phi\rangle = 1$), which assigns a (complex) weight to each configuration in $\{0, \ldots, N-1\}$. The "computational basis" for $\mathcal{Q}$ is $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$, where $|i\rangle$ assigns weight 1 to configuration i, and

weight 0 to any other configuration $j \neq i$. By definition, $\{|0\rangle, |1\rangle, \ldots, |N-1\rangle\}$ forms an orthonormal basis for $\mathcal{H}_N$, and any pure state $|\phi\rangle$ can be written as $|\phi\rangle = \sum_i \alpha_i |i\rangle$, where $\sum_i |\alpha_i|^2 = 1$. Given two quantum systems $\mathcal{Q}_0$ and $\mathcal{Q}_1$ over $\mathcal{H}_N$ and $\mathcal{H}_M$, respectively, the joint quantum system is defined via the tensor product $\mathcal{H}_N \otimes \mathcal{H}_M$, and the product state of $|\phi\rangle_0 \in \mathcal{H}_N$ and $|\phi\rangle_1 \in \mathcal{H}_M$ is denoted by $|\phi\rangle_0 \otimes |\phi\rangle_1$, or simply $|\phi_0, \phi_1\rangle$.

A qubit is a quantum system with $N = 2$ configurations labeled by $\{0, 1\}$. An n-qubit system is the joint quantum system of n qubits. The standard computational basis $\{|x\rangle\}_{x\in\{0,1\}^n}$ for an n-qubit system is given by $|x_1\rangle \otimes \cdots \otimes |x_n\rangle$, where $x = x_1 \ldots x_n$. Any classical bit-string $x \in \{0, 1\}^n$ can be encoded into a quantum state $|x\rangle$, and any arbitrary pure n-qubit state $|\phi\rangle$ can be written as $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$, where $\sum_{x\in\mathcal{X}} |\alpha_x|^2 = 1$. A pure state $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$ is in *superposition* if $|\alpha_x|^2 < 1$ for all $x \in \{0, 1\}^n$.

Quantum Measurement. Let $B = \{|x\rangle\}_{x\in\mathcal{X}}$ be any orthonormal basis of $\mathcal{H}_N$. Given a pure state $|\phi\rangle \in \mathcal{H}_N$, we can measure it in the basis B, obtaining a value x with probability $|\langle x|\phi\rangle|^2$. This operation induces a probability distribution $D_\phi(x) = |\langle x|\phi\rangle|^2$ over $\mathcal{X}$. After measurement, the state $|\phi\rangle$ collapses to $|x\rangle$, which will not change under subsequent measurements in the same basis B (but may still change under measurements in other basis). We can also perform partial measurement on a pure state in a joint quantum system. Formally, let $|\phi\rangle \in \mathcal{H}_N \otimes \mathcal{H}_M$ be a pure state of the joint quantum system $\mathcal{Q} = \mathcal{Q}_0 \otimes \mathcal{Q}_1$, and let $B_0 = \{|x\rangle\}_{x\in\mathcal{X}}$ (resp., $B_1 = \{|y\rangle\}_{y\in\mathcal{Y}}$) be an orthonormal basis of $\mathcal{H}_N$ (resp., $\mathcal{H}_M$). After a partial measurement on $|\phi\rangle$ in the basis B_0, we will obtain a value $x \in \mathcal{X}$ with probability $p_x = \sum_{y\in\mathcal{Y}} |\langle x, y|\phi\rangle|^2$, and the state $|\phi\rangle = \sum_{x\in\mathcal{X},y\in\mathcal{Y}} \alpha_{x,y} |x, y\rangle$ collapses to $\sum_{y\in\mathcal{Y}} \frac{\alpha_{x,y}}{\sqrt{p_x}} |x, y\rangle$.

Quantum Algorithms. A quantum algorithm $\mathcal{A}$ over a Hilbert space $\mathcal{H}_N$ with an orthonormal basis $\{|x\rangle\}_{x\in\mathcal{X}}$ is specified by a unitary transformation U, which takes an initial state $|\phi\rangle$ as input, and outputs a result obtained by performing a measurement on the final state $|\psi\rangle = U|\phi\rangle$. We say that a quantum algorithm $\mathcal{A}$ is efficient if U is composed of a polynomial number of universal basis gates (e.g., the Hadamard, CNOT, and $\pi/8$ gates). Let $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ be any sets such that $\mathcal{Y}$ is the additive group of $\mathbb{Z}_2^\ell$ for some $\ell \in \mathbb{N}$. Let $B = \{|x, y, z\rangle\}_{x\in\mathcal{X},y\in\mathcal{Y},z\in\mathcal{Z}}$ be the corresponding orthonormal basis. For any function $f : \mathcal{X} \to \mathcal{Y}$, define U_f as the unitary transformation that maps $|x, y, z\rangle$ into $|x, y \oplus f(x), z\rangle$. By definition, the inverse transformation of U_f is itself. Let Id be the identity transformation.

Let O_f be an oracle that computes the unitary transformation U_f. An oracle quantum algorithm A^{O_f} with at most q queries to O_f is specified by a sequence of $q+1$ unitary transformation $U_0, \cdots, U_q$. Specifically, given an initial state $|\phi\rangle$, the algorithm A^{O_f} outputs a result obtained by performing a measurement on the final state $|\psi\rangle = U_q U_f U_{q-1} \ldots U_f U_0 |\phi\rangle$. Similarly, one can define oracle algorithm $A^{O_{f_1},\ldots,O_{f_k}}$ with access to a collection of oracles $\{O_{f_1}, \ldots, O_{f_k}\}$, which is polynomially equivalent to an oracle algorithm B^O with access to a single oracle $O(k, x) = O_{f_k}(x)$. The following lemma is implicit in [34].

Lemma 1 (Algorithmic One-way to Hiding [34,20]). *Let $\mathcal{F}$ be the family of functions from $\mathcal{X}$ to $\mathcal{Y}$, and let $\mathcal{O} : \mathcal{X} \to \mathcal{Y}$ be a random oracle. Let D be any arbitrary probability distribution over $\mathcal{X}$. Let E be any arbitrary (probabilistic) algorithm which takes a pair $(x, y) \in \mathcal{X} \times \mathcal{Y}$ as inputs, outputs a bit string $inp \in \{0, 1\}^*$. Consider an oracle algorithm $\mathcal{A}$ that makes at most q queries to $\mathcal{O}$. Let $\mathcal{B}$ be an oracle algorithm that on input $inp \in \{0, 1\}^*$ does the following: pick $i \xleftarrow{\$} \{1, \ldots, q\}$, run $\mathcal{A}^{\mathcal{O}}(inp)$ until (just receiving) the i-th query, measure the argument of the query in the computational basis, output the measurement outcome. (When $\mathcal{A}$ makes less than i queries, $\mathcal{B}$ outputs $\perp$.) Let*

$$P^1_{\mathcal{A}} := \Pr\left[b' = 1 : \mathcal{O} \xleftarrow{\$} \mathcal{F}, x \xleftarrow{\$} D, y = \mathcal{O}(x), inp \leftarrow E(x, y), b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(inp)\right]$$

$$P^2_{\mathcal{A}} := \Pr\left[b' = 1 : \mathcal{O} \xleftarrow{\$} \mathcal{F}, x \xleftarrow{\$} D,\ y \xleftarrow{\$} \mathcal{Y},\ \ inp \leftarrow E(x, y), b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}}(inp)\right]$$

$$P_{\mathcal{B}} := \Pr\left[x = x' : \mathcal{O} \xleftarrow{\$} \mathcal{F}, x \xleftarrow{\$} D,\ y \xleftarrow{\$} \mathcal{Y},\ \ inp \leftarrow E(x, y), x' \xleftarrow{\$} \mathcal{B}^{\mathcal{O}}(inp)\right]$$

Then $|P^1_{\mathcal{A}} - P^2_{\mathcal{A}}| \leq 2q\sqrt{P_{\mathcal{B}}}$.

We clarify that the original one-way to hiding lemma in [34,20] only considers the uniform distribution, i.e., $x \xleftarrow{\$} \mathcal{X}$, but the proof given in [34] essentially applies to any distribution D. Here is a brief explanation:

if $\mathcal{A}$ does not use x in any RO queries, then it cannot distinguish $(x, \mathcal{O}(x))$ from the pair (x, y) with random $y \xleftarrow{\$} \mathcal{Y}$ (i.e., we already have $|P^1_{\mathcal{A}} - P^2_{\mathcal{A}}| \leq \mathrm{negl}(\kappa)$), while if $\mathcal{A}$ uses x in some RO queries, then $\mathcal{B}$ can find x by randomly choosing and measuring one of the queries from $\mathcal{A}$. This fact is essentially independent from the choice of x (note that x is always chosen from the same distribution D in Lemma 1).

2.2 Indifferentiability

As a generalization of indistinguishability, the notion of indifferentiability was introduced by Maurer et al. [25] to deal with the setting where each primitive is assumed to have two interfaces (which may be different or not): *private* and *public*, modeling the access of the honest parties and the adversary, respectively. We adapt the notion of indifferentiability to the quantum setting. Formally, a primitive $\mathcal{I} = (\mathcal{I}_{priv}(\cdot), \mathcal{I}_{pub}(\cdot))$ is said to be (resp., quantum computationally) indifferentiable from another primitive $\mathcal{J} = (\mathcal{J}_{priv}(\cdot), \mathcal{J}_{pub}(\cdot))$ if for all (resp., QPT) distinguisher $\mathcal{D}(\cdot)$, there is a (resp., QPT) algorithm $\mathcal{P}(\cdot)$ such that

$$\left|\Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{I}_{priv}(\cdot),\mathcal{I}_{pub}(\cdot)} = 1\right] - \Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{J}_{priv}(\cdot),\mathcal{P}(\cdot)^{\mathcal{J}_{pub}(\cdot)}} = 1\right]\right| \leq \mathrm{negl}(\kappa)$$

holds, where κ is the security parameter. Note that, unlike indistinguishability, indifferentiability is not symmetric, i.e., the fact that $\mathcal{I}$ is indifferentiable from $\mathcal{J}$ does not necessary imply that $\mathcal{J}$ is indifferentiable from $\mathcal{I}$. However, if the primitives have no public interfaces, then both notions are equivalent.

Definition 1 ([25]). *A cryptosystem $\mathcal{U} = (\mathcal{U}_{priv}(\cdot), \mathcal{U}_{pub}(\cdot))$ is said to be (resp., computationally) at least as secure as another cryptosystem $\mathcal{V} = (\mathcal{V}_{priv}(\cdot), \mathcal{V}_{pub}(\cdot))$ if for all (resp., QPT) algorithm $\mathcal{D}_q$ the following holds: For any (resp., QPT) adversary $\mathcal{A}$ accessing $\mathcal{U}$ there is another adversary $\mathcal{B}$ accessing $\mathcal{V}$, we have that*

$$\left|\Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{U}_{priv}(\cdot),\mathcal{A}(\cdot)^{\mathcal{U}_{pub}(\cdot)}} = 1\right] - \Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{V}_{priv}(\cdot),\mathcal{B}(\cdot)^{\mathcal{V}_{pub}(\cdot)}} = 1\right]\right| \leq \mathrm{negl}(\kappa).$$

By definition the public interface of a primitive is always available to the adversary. In particular, if $\mathcal{C}(\mathcal{I})$ is a cryptosystem which uses primitive $\mathcal{I}$ as a component, the public interface $\mathcal{I}_{pub}(\cdot)$ of $\mathcal{I} = (\mathcal{I}_{priv}(\cdot), \mathcal{I}_{pub}(\cdot))$ is still available to the adversary attacking $\mathcal{C}(\mathcal{I})$. This is different from the notion of indistinguishability, where an adversary attacking a system $\mathcal{C}(\cdot)$ is not allowed to directly access its building blocks. The following lemma is obtained by directly adapting [25, Theorem 1] to the quantum setting (since [25, Theorem 1] is proved by simply renaming the interfaces of the involved systems).

Lemma 2 ([25]). *Let $\mathcal{I} = (\mathcal{I}_{priv}(\cdot), \mathcal{I}_{pub}(\cdot))$ and $\mathcal{J} = (\mathcal{J}_{priv}(\cdot), \mathcal{J}_{pub}(\cdot))$ be two arbitrary primitives. Then, $\mathcal{I}$ is (resp., quantum computationally) indifferentiable from $\mathcal{J}$ if and only if for all (resp., QPT) construction $\mathcal{C}(\cdot)$, we have that $\mathcal{C}(\mathcal{I})$ is (resp., quantum computationally) at least as secure as $\mathcal{C}(\mathcal{J})$.*

3 Separations of the (Quantum) Random Oracle Model

Note that the only difference between the ROM and the QROM is how the adversary uses the RO: the adversary in the ROM only makes classical queries to the RO, but that in the QROM can make quantum queries to the RO. This fact motivates us to formalize the (Q)ROM by using the indifferentiability framework in [25], and equipping the RO with a *private* and a *public* interface.

Let n, m be positive integers, and let $\mathcal{F}_{n,m} = \{f : \{0,1\}^n \to \{0,1\}^m\}$ be a family of functions. A RO $\mathcal{O}(\cdot)$ from $\{0,1\}^n$ to $\{0,1\}^m$ is a "black-box" random function uniformly chosen from $\mathcal{F}_{n,m}$, which can only be accessed via given interfaces. We say that $\mathcal{O}(\cdot) \xleftarrow{\$} \mathcal{F}_{n,m}$ is a classical RO (CRO) if it provides a private interface $\mathcal{O}_{priv}(\cdot)$ for the honest parties and an essentially the same public interface $\mathcal{O}_{pub}(\cdot)$ for the adversary (i.e., $\mathcal{O}_{pub}(\cdot) = \mathcal{O}_{priv}(\cdot) = \mathcal{O}(\cdot)$).

In contrast, we say that $\mathcal{O}(\cdot) \xleftarrow{\$} \mathcal{F}_{n,m}$ is a quantum RO (QRO) if it provides two different interfaces $\mathcal{O}_{priv}(\cdot)$ and $\mathcal{O}_{pub}(\cdot)$, where the private interface $\mathcal{O}_{priv}(\cdot)$ allows the honest parties to classically access $\mathcal{O}(\cdot)$ and returns $\mathcal{O}_{priv}(x) = \mathcal{O}(x)$ to a classical query $x \in \{0,1\}^n$, while the public interface $\mathcal{O}_{pub}(\cdot)$ enables the

adversary to quantumly access $\mathcal{O}(\cdot)$ and returns a state $\mathcal{O}_{pub}(|x,z\rangle) = |x, z \oplus \mathcal{O}(x)\rangle$ to a quantum query $|x,z\rangle$ of $n+m$ qubits.

We will simply denote a RO as $\mathcal{O} = (\mathcal{O}_{priv}(\cdot), \mathcal{O}_{pub}(\cdot))$ or $\mathcal{O}(\cdot)$ if we do not distinguish whether it is classical or quantum. Otherwise, we directly denote a classical RO (CRO) as $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot))$ and a quantum RO (QRO) as $\mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot))$ by using superscripts "c" and "q". In this formalization, a cryptosystem in the (Q)ROM is a (quantum) polynomial time construction which uses a (QRO) CRO as a component.

3.1 The QROM is Stronger than the ROM

By Lemma 2, it suffices to show that the CRO is indifferentiable from the QRO, but the reverse does not hold. We will do this by using Theorem 1 and Theorem 2.

Theorem 1. *Let $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$ be a CRO, and let $\mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$ be a QRO. Then, $\mathcal{O}^c$ is indifferentiable from $\mathcal{O}^q$. In particular, for all (even unbounded) distinguisher $\mathcal{D}_q$, there exists a QPT algorithm $\mathcal{P}$ such that*

$$\left|\Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{O}^c_{priv}(\cdot),\mathcal{O}^c_{pub}(\cdot)} = 1\right] - \Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{O}^q_{priv}(\cdot),\mathcal{P}(\cdot)^{\mathcal{O}^q_{pub}(\cdot)}} = 1\right]\right| = 0.$$

Proof. Note that the private interfaces of both $\mathcal{O}^c$ and $\mathcal{O}^q$ models the access of the honest parties to a "black-box" random function, and essentially have identical behaviors. Moreover, the public interface $\mathcal{O}^c_{pub}(\cdot)$ of $\mathcal{O}^c$ has the same behavior as its private interface $\mathcal{O}^c_{priv}(\cdot)$ by definition. To finish the proof, we construct a QPT oracle algorithm $\mathcal{P}(\cdot)$ which receives a classical query x, and returns a response y such that it is consistent with the one obtained by sending x to $\mathcal{O}^q(\cdot)$ via the private interface $\mathcal{O}^q_{priv}(\cdot)$. Specifically, when receiving a query $x \in \{0,1\}^n$ from the distinguisher $\mathcal{D}_q$, the algorithm $\mathcal{P}(\cdot)$ performs the followings:

- Prepare a state $|x, 0_m\rangle$ of $n+m$ qubits;
- Send a quantum query $|x, 0_m\rangle$ to its own oracle $\mathcal{O}^q(\cdot)$ via the public interface $\mathcal{O}^q_{pub}(\cdot)$, which will return a state $|x, \mathcal{O}^q(x)\rangle$ by definition;
- Measure the state $|x, \mathcal{O}^q(x)\rangle$ to obtain $\hat{x} \in \{0,1\}^n$ and $\hat{y} = \mathcal{O}^q(\hat{x})$. Return $\hat{y} \in \{0,1\}^m$ to the distinguisher $\mathcal{D}_q$.

By definition, $\hat{y}$ is consistent with the response obtained by querying $\hat{x}$ to $\mathcal{O}^q(\cdot)$ via the private interface $\mathcal{O}^q_{priv}(\cdot)$. Thus, both $\mathcal{O}^q_{priv}(\cdot)$ and $\mathcal{P}(\cdot)^{\mathcal{O}^q_{pub}(\cdot)}$ have the same behaviors in the distinguisher $\mathcal{D}_q$'s view, which is identical to the case when $\mathcal{D}_q$ is given access to $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot))$. This completes the proof. □

At first glance, one might think that a "reversal" variant (which first measures the quantum query, and then uses the outcome to make a classical query) of the algorithm $\mathcal{P}(\cdot)$ in the proof of Theorem 1 suffices to show that $\mathcal{O}^q$ is indifferentiable from $\mathcal{O}^c$. Unfortunately, we show that any (even unbounded) algorithm making only a polynomial number of classical queries will not work.

Theorem 2. *Let κ be the security parameter. Let $n \geq \omega(\log \kappa)$ and $m \geq 1$ be two integers. Let $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$ be a CRO, and let $\mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$ be a QRO. Then, $\mathcal{O}^q$ is differentiable from $\mathcal{O}^c$. In particular, for sufficiently large n, there exists a QPT distinguisher $\mathcal{D}_q$ such that for all (even unbounded) quantum algorithm $\mathcal{P}(\cdot)$ making only a polynomial number ℓ of classical queries to its own oracle, we have that*

$$\left|\Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{O}^q_{priv}(\cdot),\mathcal{O}^q_{pub}(\cdot)} = 1\right] - \Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{O}^c_{priv}(\cdot),\mathcal{P}(\cdot)^{\mathcal{O}^c_{pub}(\cdot)}} = 1\right]\right| \geq \frac{1}{72}.$$

Note that Theorem 2 directly rules out the trivial distinguisher which runs the Grover algorithm [19,11] with sub-exponentially many, i.e., $O(2^{n/2})$ or $O(2^{m/3})$, quantum queries, as it is useless for most schemes

that are based on quantum sub-exponential time (or even polynomial time) hardness assumptions. Actually, we will construct a QPT distinguisher $\mathcal{D}_q$ making at most two quantum queries.

Before proceeding to prove Theorem 2, we first give an information versus disturbance lemma, which might be of independent interest. Formally, let H^1 be the one-dimensional Hadamard transformation H (i.e., $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$), and let H^0 be the identity transformation Id. Then, for any $s \in \{0,1\}^n$, we can define a quantum operation $H^s \stackrel{\text{def}}{=} H^{s_1} \otimes H^{s_2} \cdots \otimes H^{s_n}$ on n qubits: for any bit string $x \in \{0,1\}^n$ and $s \in \{0,1\}^n$, denote $|x\rangle_s \stackrel{\text{def}}{=} H^s|x\rangle$ as the quantum encoding $|x\rangle_s = |x_1\rangle_{s_1} \cdots |x_n\rangle_{s_n}$ of $x = x_1 \dots x_n$ by using bases indicated by $s = s_1 \dots s_n$, where $|x_i\rangle_{s_i} \stackrel{\text{def}}{=} |x_i\rangle$ if $s_i = 0$, and $|x_i\rangle_{s_i} \stackrel{\text{def}}{=} H|x_i\rangle$ otherwise. Clearly, we always have $H^s|x\rangle_s = |x\rangle$.

Lemma 3 (Information versus Disturbance). *Let x, s be two random variables uniformly distributed over $\{0,1\}^n$. Let $\mathcal{P}$ be any (unbounded) quantum algorithm which takes $|x\rangle_s$ as input, outputs a bit string $e \in \{0,1\}^{\mathrm{poly}(n)}$ and a quantum state $|\zeta\rangle$ of n-qubit, i.e., $(e, |\zeta\rangle) \leftarrow \mathcal{P}(|x\rangle_s)$. Let $x' \in \{0,1\}^n$ be a bit string obtained by first applying H^s on the state $|\zeta\rangle$, and then measuring the state $H^s|\zeta\rangle$ in the computational basis. Let $z = x \oplus x'$, and let $I(x,e)$ be the mutual information between x and e. Then, we have that*

$$I(x,e) \leq n(\alpha + \frac{1}{\alpha} \sum_{\mathsf{hw}(z) \geq 1} \Pr[z])$$

holds for any $\alpha > 0$, where $\mathsf{hw}(z)$ denotes the hamming-weight of $z \in \{0,1\}^n$.

Note that if $|\zeta\rangle = |x\rangle_s$, we always have $z = 0_n$ (i.e., $\mathsf{hw}(z) = 0$). The above lemma is basically a quantitative version of the claim that any attempt made by $\mathcal{P}$ to obtain useful information x from the state $|x\rangle_s$ will necessarily disturb the state. The proof of Lemma 3 is essentially adapted from the security proof [6] of BB84 protocol. Actually, one can imagine that there are two users "Alice" and "Bob" involved in the interactions with $\mathcal{P}$, where "Alice" prepares the input state to $\mathcal{P}$, and "Bob" checks if the input state is disturbed by $\mathcal{P}$. Moreover, the behaviors of "Alice" and "Bob" are similar to those of the real users in the used-bits-BB84 protocol (before the information reconciliation procedure) [6]. The major difference is that in our case "Alice" and "Bob" share the same random $x, s \in \{0,1\}^n$ (i.e., they will not leak any classical information of x, s to the "eavesdropper" $\mathcal{P}$), which allows us to obtain a "clean" information versus disturbance lemma (note that the eavesdropper in the BB84 protocol can also obtain information from a classical channel, which makes a similar result [6] for the BB84 protocol more complex and unsuitable for our purpose).

Proof. Without loss of generality, we can assume that $\mathcal{P}$ works as follows: given an input state $|x\rangle_s$ in the input register, it first prepares an ancillary register A in a known state $|0\rangle_A$, and performs a unitary transformation U on the state

$$|0\rangle_A |x\rangle_s .$$

The resulting state $U|0\rangle_A |x\rangle_s$ can be expressed in a unique way as a sum

$$U|0\rangle_A |x\rangle_s = \sum_{\hat{x}} |U_{x,\hat{x}}\rangle_s |\hat{x}\rangle_s ,$$

where $|U_{x,\hat{x}}\rangle_s = {}_s\langle\hat{x}|U|0\rangle_A |x\rangle_s$ are non-normalized states of $\mathcal{P}$'s register A. Then, it performs some measurement M on the state in his ancillary register to obtain a classical information e and a "disturbed" state $|\zeta\rangle$ in the input register. Finally, it outputs e and $|\zeta\rangle$.

Let x' be the measurement outcome of $H^s|\zeta\rangle$. Let $z = x \oplus x'$. Then, we have

$$\Pr[z] = \sum_{x,s} \Pr[x,s] \Pr[z|x,s] = \frac{1}{2^{2n}} \sum_{x,s} \langle U_{x,x\oplus z} | U_{x,x\oplus z} \rangle_s . \quad (1)$$

Note that $\mathcal{P}$'s state in the ancillary register A after performing the unitary transformation U is fully determined by tracing-out the subsystem $|\hat{x}\rangle_s$ from the state $\sum_{\hat{x}} |U_{x,\hat{x}}\rangle_s |\hat{x}\rangle_s$, and it is

$$\rho_s^x = \sum_{\hat{x}} |U_{x,\hat{x}}\rangle_s \, {}_s\langle U_{x,\hat{x}}|.$$

We can purify the above state while giving more information to $\mathcal{P}$ by assuming she keeps the pure state

$$|\phi_x\rangle_s = \sum_{\hat{x}} |U_{x,\hat{x}}\rangle_s |x \oplus \hat{x}\rangle_s .$$

We have that $\rho_s^x = |\phi_x\rangle_s \, {}_s\langle\phi_x|$.

As shown in [6], it is sufficient to consider the symmetric attack, which is irrelevant to and will not be affected by the choices of $x \in \{0,1\}^n$ and $s \in \{0,1\}^n$. In fact, Biham et al. [6] showed that for any attack $\{U, M\}$, one can define a symmetric attack $\{U^{sym}, M^{sym}\}$ which is at least as good (for $\mathcal{P}$) as the original attack $\{U, M\}$. In particular, compared to the original attack, the symmetric one does not decrease the information obtained by $\mathcal{P}$ while keeping the same average error-rate caused by $\mathcal{P}$. For symmetric attack $\{U, M\}$, we have the following useful facts [6, Lemma 3.5]:

- $\langle U_{x,x\oplus z}|U_{x\oplus t,x\oplus z\oplus t}\rangle_s$ is independent of x;
- $\sum_{\hat{x}} \langle U_{x,\hat{x}}|U_{x\oplus t,\hat{x}\oplus t}\rangle_s$ is independent of x.

Define $\Phi_{t,s} \stackrel{\text{def}}{=} \langle\phi_x|\phi_{x\oplus t}\rangle_s = \sum_{\hat{x}} \langle U_{x,\hat{x}}|U_{x\oplus t,\hat{x}\oplus t}\rangle_s$, which is independent of x. Define

$$|\gamma_x\rangle_s \stackrel{\text{def}}{=} \frac{1}{2^n}\sum_t (-1)^{x\cdot t} |\phi_t\rangle_s , \quad d_{x,s}^2 \stackrel{\text{def}}{=} \langle\gamma_x|\gamma_x\rangle_s \text{ and } \hat{\gamma}_x \stackrel{\text{def}}{=} \gamma_x/d_{x,s},$$

where $x \cdot t = (x_1 \cdot t_1) \oplus \cdots \oplus (x_n \cdot t_n) \in \{0,1\}$ for any bit vector $x = (x_1, \ldots, x_n), t = (t_1, \ldots, t_n) \in \{0,1\}^n$ and $d_{x,s} > 0$. We now slightly deviate from the main proof by showing several useful equations which will be used latter.

Firstly, by the fact that

$$\frac{1}{2^n}\sum_t (-1)^{(x\oplus\hat{x})\cdot t} = \begin{cases} 0, x \neq \hat{x}; \\ 1, \text{otherwise}, \end{cases}$$

we can rewrite

$$|\phi_t\rangle_s = \sum_{\hat{x}} (-1)^{\hat{x}\cdot t} |\gamma_{\hat{x}}\rangle_s = \sum_{\hat{x}} (-1)^{\hat{x}\cdot t} d_{\hat{x},s} |\hat{\gamma}_{\hat{x}}\rangle_s \tag{2}$$

Secondly, we can rewrite the equation $\langle\gamma_x|\gamma_x\rangle_s = \frac{1}{2^{2n}}\sum_{t,\hat{t}} (-1)^{x\cdot(t\oplus\hat{t})} \langle\phi_t|\phi_{\hat{t}}\rangle_s$ as $\langle\gamma_x|\gamma_x\rangle_s = \frac{1}{2^{2n}}\sum_{t,\hat{t}} (-1)^{x\cdot\hat{t}} \langle\phi_t|\phi_{t\oplus\hat{t}}\rangle_s$ by using variable substitution, which in turn implies that

$$d_{x,s}^2 = \frac{1}{2^{2n}}\sum_{t,\hat{t}} (-1)^{x\cdot\hat{t}} \langle\phi_t|\phi_{t\oplus\hat{t}}\rangle_s = \frac{1}{2^{2n}}\sum_{t,\hat{t},\hat{x}} (-1)^{x\cdot\hat{t}} \langle U_{t,\hat{x}}|U_{t\oplus\hat{t},\hat{x}\oplus\hat{t}}\rangle_s . \tag{3}$$

Thirdly, for any $x \neq \hat{x}$, we have

$$\begin{aligned} \langle\gamma_x|\gamma_{\hat{x}}\rangle_s &= \tfrac{1}{2^{2n}}\textstyle\sum_{t,\hat{t}} (-1)^{x\cdot t}(-1)^{\hat{x}\cdot\hat{t}} \langle\phi_t|\phi_{\hat{t}}\rangle_s \\ &= \tfrac{1}{2^{2n}}\textstyle\sum_{t,\hat{t}} (-1)^{(x\oplus\hat{x})\cdot t}(-1)^{\hat{x}\cdot\hat{t}} \langle\phi_t|\phi_{t\oplus\hat{t}}\rangle_s \\ &= \tfrac{1}{2^{2n}}\textstyle\sum_{t,\hat{t}} (-1)^{(x\oplus\hat{x})\cdot t}(-1)^{\hat{x}\cdot\hat{t}} \Phi_{\hat{t},s} \\ &= \tfrac{1}{2^{2n}}(\textstyle\sum_t (-1)^{(x\oplus\hat{x})\cdot t}) \sum_{\hat{t}} (-1)^{\hat{x}\cdot\hat{t}} \Phi_{\hat{t},s}. \end{aligned}$$

Since $\sum_t (-1)^{(x\oplus\hat{x})\cdot t} = 0$, we have that

$$\langle\gamma_x|\gamma_{\hat{x}}\rangle_s = 0 \tag{4}$$

holds for any $x \neq \hat{x}$.

Fourthly, by the fact that $\sum_x (-1)^{x\cdot\hat{t}} = 0$ for any $\hat{t} \neq 0_n$, and that $|\phi_t\rangle$ is a pure state (which implies $\langle\phi_t|\phi_t\rangle_s = 1$ for any $t \in \{0,1\}^n$), we have that

$$\begin{aligned}\sum_x d_{x,s}^2 &= \sum_x \tfrac{1}{2^{2n}} \sum_{t,\hat{t}} (-1)^{x\cdot\hat{t}} \left\langle \phi_t \middle| \phi_{t\oplus\hat{t}} \right\rangle_s \\ &= \tfrac{1}{2^{2n}} \sum_{t,\hat{t}} \left(\sum_x (-1)^{x\cdot\hat{t}} \right) \left\langle \phi_t \middle| \phi_{t\oplus\hat{t}} \right\rangle_s \\ &= \tfrac{1}{2^n} \sum_t \langle \phi_t | \phi_t \rangle_s \\ &= 1.\end{aligned} \tag{5}$$

Next, we return back to the main proof. Let $I(x,e)$ be the mutual information of x and e. Let $I(x_i,e)$ be the mutual information between the i-th bit x_i of x and e, where $i \in \{1,\ldots,n\}$. Since x is a random variable uniformly distributed over $\{0,1\}^n$, we have that

$$I(x,e) \leq \sum_i I(x_i,e). \tag{6}$$

Let $v_i \in \{0,1\}^n$ be the bit string whose j-th bit is nonzero if and only if $j = i$, we have $x_i = v_i \cdot x \in \{0,1\}$. For any bit $a \in \{0,1\}$, define

$$\begin{aligned}\rho_a(v_i) &= \tfrac{1}{2^{2n-1}} \sum_s \sum_{v_i\cdot x = a} \rho_s^x \\ &= \tfrac{1}{2^{2n-1}} \sum_s \sum_{v_i\cdot x = a} |\phi_x\rangle_{s\ s} \langle \phi_x| \\ &= \tfrac{1}{2^{2n-1}} \sum_{s,t,\hat{t}} \sum_{v_i\cdot x = a} (-1)^{(t\oplus\hat{t})\cdot x} d_{t,s} d_{\hat{t},s} \, |\hat{\gamma}_t\rangle_{s\ s} \langle \hat{\gamma}_{\hat{t}}|,\end{aligned}$$

where the last equation is due to Equation (2). Note that in order to distinguish the i-th bit x_i of x, $\mathcal{P}$ has to distinguish the two states $\rho_0(v_i)$ and $\rho_1(v_i)$. A good measure for the distinguishability of $\rho_0(v_i)$ and $\rho_1(v_i)$ is the optimal mutual information that one could get if one needs to guess the bit a by performing an optimal measurement to distinguish between the two density matrices, when the two are given with equal probability of half. This information is called the Shannon Distinguishability [15], denote as $SD(\rho_0(v_i), \rho_1(v_i))$. Due to the optimality of SD, we get

$$I(x_i,e) \leq SD(\rho_0(v_i), \rho_1(v_i)),$$

which is then bounded by the trace norm of $\rho_0(v_i) - \rho_1(v_i)$ [15]. Since

$$\begin{aligned}\rho_0(v_i) - \rho_1(v_i) &= (-1)^0 \rho_0(v_i) + (-1)^1 \rho_1(v_i) \\ &= \tfrac{1}{2^{2n-1}} \sum_{s,x,t,\hat{t}} (-1)^{(t\oplus\hat{t}\oplus v_i)\cdot x} d_{t,s} d_{\hat{t},s} \, |\hat{\gamma}_t\rangle_{s\ s} \langle \hat{\gamma}_{\hat{t}}| \\ &= \tfrac{1}{2^{2n-1}} \sum_{s,t,\hat{t}} \left(\sum_x (-1)^{(t\oplus\hat{t}\oplus v_i)\cdot x} \right) d_{t,s} d_{\hat{t},s} \, |\hat{\gamma}_t\rangle_{s\ s} \langle \hat{\gamma}_{\hat{t}}| \\ &= \tfrac{1}{2^{n-1}} \sum_{s,t} d_{t,s} d_{t\oplus v_i,s} \, |\hat{\gamma}_t\rangle_{s\ s} \langle \hat{\gamma}_{t\oplus v_i}|,\end{aligned}$$

we have

$$\begin{aligned}SD(\rho_0(v_i), \rho_1(v_i)) &\leq \tfrac{1}{2} Tr |\rho_0(v_i) - \rho_1(v_i)| \\ &\leq \tfrac{1}{2^n} Tr \left| \sum_{s,t} d_{t,s} d_{t\oplus v_i,s} \, |\hat{\gamma}_t\rangle_{s\ s} \langle \hat{\gamma}_{t\oplus v_i}| \right| \\ &= \tfrac{1}{2^{n+1}} Tr \left| \sum_{s,t} d_{t,s} d_{t\oplus v_i,s} (|\hat{\gamma}_t\rangle_{s\ s} \langle \hat{\gamma}_{t\oplus v_i}| + |\hat{\gamma}_{t\oplus v_i}\rangle_s \langle \hat{\gamma}_t|_s) \right| \\ &\leq \tfrac{1}{2^n} \sum_{s,t} d_{t,s} d_{t\oplus v_i,s} (\tfrac{1}{2} Tr |\, |\hat{\gamma}_t\rangle_s \langle \hat{\gamma}_{t\oplus v_i}|_s + |\hat{\gamma}_{t\oplus v_i}\rangle_{s\ s} \langle \hat{\gamma}_t||) \\ &= \tfrac{1}{2^n} \sum_{s,t} d_{t,s} d_{t\oplus v_i,s} \sqrt{1 - (\mathrm{Im}(\langle \hat{\gamma}_t | \hat{\gamma}_{t\oplus v_i} \rangle_s))^2} \\ &= \tfrac{1}{2^n} \sum_{s,t} d_{t,s} d_{t\oplus v_i,s} \\ &= \tfrac{1}{2^n} \sum_s \left(\sum_{\mathsf{hw}(t)\geq 1} d_{t,s} d_{t\oplus v_i,s} + \sum_{\mathsf{hw}(t)=0} d_{t,s} d_{t\oplus v_i,s} \right) \\ &\leq \tfrac{1}{2^n} \sum_s \left(\sum_{\mathsf{hw}(t)\geq 1} d_{t,s} d_{t\oplus v_i,s} + \sum_{\mathsf{hw}(t\oplus v_i)\geq 1} d_{t,s} d_{t\oplus v_i,s} \right),\end{aligned}$$

where Im is the imaginary part, and the last third equality holds due to the fact that $\langle\hat{\gamma}_t|\hat{\gamma}_{t\oplus v_i}\rangle_s = 0$ by Equation (4). For any positive $\alpha > 0$, we have

$$\begin{aligned}
SD(\rho_0(v_i), \rho_1(v_i)) &\leq \tfrac{1}{2^n}\textstyle\sum_s \left(2\sum_{\mathsf{hw}(t)\geq 1} d_{t,s}d_{t\oplus v_i,s}\right) \\
&= \tfrac{1}{2^n}\textstyle\sum_s \left(\frac{1}{\alpha}\sum_{\mathsf{hw}(t)\geq 1} 2\alpha d_{t,s}d_{t\oplus v_i,s}\right) \\
&\leq \tfrac{1}{2^n}\textstyle\sum_s \left(\frac{1}{\alpha}\sum_{\mathsf{hw}(t)\geq 1} (d_{t,s}^2 + \alpha^2 d_{t\oplus v_i,s}^2)\right) \\
&= \tfrac{1}{2^n}\textstyle\sum_s \left(\frac{1}{\alpha}\sum_{\mathsf{hw}(t)\geq 1} d_{t,s}^2 + \alpha\sum_{\mathsf{hw}(t)\geq 1} d_{t\oplus v_i,s}^2\right) \\
&\leq \tfrac{1}{2^n}\textstyle\sum_s \left(\frac{1}{\alpha}\sum_{\mathsf{hw}(t)\geq 1} d_{t,s}^2 + \alpha\right)
\end{aligned}$$

The third inequality follows from the Cauchy-Schwarz inequality, and the last one holds because $\sum_{\mathsf{hw}(t)\geq 1} d_{t\oplus v_i,s}^2 \leq \sum_t d_{t,s}^2 = 1$ by Equation (5). This means that

$$I(x_i, e) \leq SD(\rho_0(v_i), \rho_1(v_i)) \leq \alpha + \frac{1}{\alpha 2^n}\sum_s \sum_{\mathsf{hw}(t)\geq 1} d_{t,s}^2$$

holds for any $\alpha > 0$.

By Equation (6), we have

$$I(x, e) \leq \sum_i I(x_i, e) \leq n\left(\alpha + \frac{1}{\alpha 2^n}\sum_s \sum_{\mathsf{hw}(t)\geq 1} d_{t,s}^2\right) \tag{7}$$

We finish the proof by bounding $I(x, e)$ using $\Pr[z]$. By Equation (1), we have

$$\Pr[z] = \sum_{x,s} \Pr[x, s]\Pr[c|x, s] = \frac{1}{2^{2n}}\sum_{x,s} \langle U_{x,x\oplus z}|U_{x,x\oplus z}\rangle_s .$$

For any $s \in \{0,1\}^n$, let $\bar{s} = s \oplus 1_n \in \{0,1\}^n$ be the bit string obtained by flipping each bit of $s \in \{0,1\}^n$. Since the change of basis between s and $\bar{s}$ is expressed by $|x'\rangle_{\bar{s}} = \sum_x 2^{-n/2}(-1)^{x'\cdot x}|x\rangle_s$ and $|x\rangle_s = \sum_{x'} 2^{-n/2}(-1)^{x\cdot x'}|x'\rangle_{\bar{s}}$, we have that

$$|U_{x',\hat{x}'}\rangle_{\bar{s}} = \frac{1}{2^n}\sum_{x,\hat{x}} (-1)^{x'\cdot x}(-1)^{\hat{x}'\cdot\hat{x}}|U_{x,\hat{x}}\rangle_s .$$

This implies that

$$\begin{aligned}
\Pr[z] &= \tfrac{1}{2^{2n}}\textstyle\sum_{t,\bar{s}} \langle U_{t,t\oplus z}|U_{t,t\oplus z}\rangle_{\bar{s}} \\
&= \tfrac{1}{2^{4n}}\textstyle\sum_{t,\bar{s}}\sum_{x,\hat{x}}\sum_{x',\hat{x}'} (-1)^{t\cdot x}(-1)^{(t\oplus z)\cdot\hat{x}}(-1)^{t\cdot x'}(-1)^{(t\oplus z)\cdot\hat{x}'} \langle U_{x,\hat{x}}|U_{x',\hat{x}'}\rangle_s \\
&= \tfrac{1}{2^{4n}}\textstyle\sum_{\bar{s},x,\hat{x},x',\hat{x}'} \left(\sum_t (-1)^{t\cdot(x\oplus x'\oplus\hat{x}\oplus\hat{x}')}\right)(-1)^{z\cdot(\hat{x}\oplus\hat{x}')} \langle U_{x,\hat{x}}|U_{x',\hat{x}'}\rangle_s .
\end{aligned}$$

Since $\sum_t (-1)^{t\cdot(x\oplus x'\oplus\hat{x}\oplus\hat{x}')} \neq 0 \Leftrightarrow x\oplus x'\oplus\hat{x}\oplus\hat{x}' = 0$, by setting $\hat{t} = x\oplus x' = \hat{x}\oplus\hat{x}'$ and using Equation (3), we have that

$$\Pr[z] = \frac{1}{2^{3n}}\sum_{\bar{s},x,\hat{x},\hat{t}} (-1)^{z\cdot\hat{t}} \left\langle U_{x,\hat{x}}\middle|U_{x\oplus\hat{t},\hat{x}\oplus\hat{t}}\right\rangle_s = \frac{1}{2^n}\sum_{\bar{s}} d_{z,s}^2 = \frac{1}{2^n}\sum_s d_{z,s}^2. \tag{8}$$

Combining Equations (7) and (8), we obtain

$$I(x, e) \leq n\left(\alpha + \frac{1}{\alpha}\sum_{\mathsf{hw}(z)\geq 1} \Pr[z]\right).$$

This completes the proof. □

Now, we are ready to prove Theorem 2. Technically, our distinguisher $\mathcal{D}_q$ will first send a quantum query such that 1) it is a superposition of a random subset $X \subseteq \{0,1\}^n$ of classical queries having the same pattern; 2) any attempt of $\mathcal{P}$ to obtain the information about X will necessarily disturb the query state; and 3) it is infeasible for $\mathcal{P}$ to make a correctly patterned classical query $x \in X$ without knowing sufficient information about X. After receiving a response from $\mathcal{P}$, $\mathcal{D}_q$ will flip a random bit to either check the consistency of the response (i.e., $\mathcal{P}$ has to make a correctly patterned classical query $x \in X$), or detect the disturbance of the query state (i.e., $\mathcal{P}$ cannot perform operations such as measurements that are not unitary) by using another quantum query. Clearly, $\mathcal{D}_q$ wins if it obtains a wrong response from the "oracle" or it detects that the "oracle" is not unitary (since both will never happen if it is given access to a real QRO).

Proof. Note that the distinguisher $\mathcal{D}_q$ is given access to an oracle $\tilde{\mathcal{O}} = (\tilde{\mathcal{O}}_{priv}(\cdot), \tilde{\mathcal{O}}_{pub}(\cdot))$, which is either a real QRO (i.e., $\tilde{\mathcal{O}} = \mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$) or a simulated one (i.e., $\tilde{\mathcal{O}} = (\mathcal{O}^c_{priv}(\cdot), \mathcal{P}(\cdot)^{\mathcal{O}^c_{pub}(\cdot)})$) for some algorithm $\mathcal{P}(\cdot)$ and CRO $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$. Moreover, given a query $x \in \{0,1\}^n$, the private interface $\tilde{\mathcal{O}}_{priv}(\cdot)$ will return a bit string $y \in \{0,1\}^m$, while given a quantum query $|x,z\rangle$, the public interface $\tilde{\mathcal{O}}_{pub}(\cdot)$ will return a state $|\psi\rangle$.

For any bit string $s = s_1 \dots s_n \in \{0,1\}^n$, define an associated set $S = \{i_1, \dots, i_{\mathsf{hw}(s)}\}$ such that $i_j \in S$ if and only if the i_j-th bit s_{i_j} of s is 1, where $\mathsf{hw}(s)$ is the Hamming weight of s and $i_1 < \cdots < i_{\mathsf{hw}(s)}$. Let $\bar{s} \in \{0,1\}^n$ be the bit string obtained by flipping each bit of s, i.e., $\bar{s} = s \oplus 1_n$. Denote $x^s = x_{i_1} \dots x_{i_{\mathsf{hw}(s)}} \in \{0,1\}^{\mathsf{hw}(s)}$ as the $\mathsf{hw}(s)$-bit substring of $x \in \{0,1\}^n$ indexed by S. Corresponding, denote $x^{\bar{s}} \in \{0,1\}^{\mathsf{hw}(\bar{s})} = \{0,1\}^{n-\mathsf{hw}(s)}$ as the substring of $x \in \{0,1\}^n$ by deleting the bits indexed by S. Now, we are ready to give the description of the distinguisher $\mathcal{D}_q$, which will make at most two quantum queries to $\tilde{\mathcal{O}}(\cdot)$ via the public interface $\tilde{\mathcal{O}}_{pub}(\cdot)$, and one classical query to $\tilde{\mathcal{O}}(\cdot)$ via the private interface $\tilde{\mathcal{O}}_{priv}(\cdot)$. Specifically, $\mathcal{D}_q$ works as follows:

1. (**Create a Random Pattern**) Uniformly choose bit-strings $x, s \xleftarrow{\$} \{0,1\}^n$ at random, and prepare a quantum state $|x\rangle|0_m\rangle$ of $n+m$ qubits. Then, apply H^s on the state $|x\rangle$ to obtain $|x\rangle_s = H^s|x\rangle$. Let $X \subseteq \{0,1\}^n$ be the set $X = \{\tilde{x} \in \{0,1\}^n : \tilde{x}^{\bar{s}} = x^{\bar{s}}\}$ determined by (x,s), we can rewrite
$$|x\rangle_s = H^s|x\rangle = \frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\tilde{x} \in X} (-1)^{x^s \cdot \tilde{x}^s} |\tilde{x}\rangle .$$
.
2. Send a quantum query $|\phi\rangle = |x\rangle_s |0_m\rangle$ to the oracle $\tilde{\mathcal{O}}(\cdot)$ via the public interface $\tilde{\mathcal{O}}_{pub}(\cdot)$, and obtain a state $|\psi\rangle$ consisting of $n+m$ qubits;
3. Pick a bit $b \xleftarrow{\$} \{0,1\}$ at random, and do the following computations:
   - 3.1 If $b = 0$, measure the state $|\psi\rangle$ to obtain a pair of $\tilde{x} \in \{0,1\}^n$ and $\tilde{y} \in \{0,1\}^m$. If $\tilde{x}^{\bar{s}} \neq x^{\bar{s}}$ (i.e., $\tilde{x} \notin X$), output 0 and abort. Otherwise, send a query $\tilde{x}$ to the oracle $\tilde{\mathcal{O}}(\cdot)$ via the private interface $\tilde{\mathcal{O}}_{priv}(\cdot)$, and obtain a result $\tilde{y}'$. If $\tilde{y}' \neq \tilde{y}$, output 0 and abort.
   - 3.2 Else if $b = 1$, send a quantum query with state $|\psi\rangle$ to the oracle $\tilde{\mathcal{O}}(\cdot)$ via the public interface $\tilde{\mathcal{O}}_{pub}(\cdot)$, and obtain a state $|\zeta\rangle$ of $n+m$ qubits. Apply H^s to the first n-qubit of the state $|\zeta\rangle$, and obtain a state $|\zeta'\rangle$. Measure the state $|\zeta'\rangle$ to obtain a pair of $\hat{x} \in \{0,1\}^n$ and $\hat{y} \in \{0,1\}^m$. If $\hat{x} \neq x$ or $\hat{y} \neq 0_m$, output 0 and abort.
4. If no abort has happened in step 3, output 1 and abort.

Now, it is enough to show the following two claims: 1) if $\tilde{\mathcal{O}}$ is a real QRO, namely, $\tilde{\mathcal{O}} = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot))$, $\mathcal{D}_q$ will always output 1; and 2) if $\tilde{\mathcal{O}}$ is a simulated one, namely, $\tilde{\mathcal{O}} = (\mathcal{O}^c_{priv}(\cdot), \mathcal{P}(\cdot)^{\mathcal{O}^c_{pub}(\cdot)})$ for some CRO $\mathcal{O}^c = (\mathcal{O}^c_{priv}(\cdot), \mathcal{O}^c_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$ and algorithm $\mathcal{P}(\cdot)$ making only a polynomial number ℓ of classical queries to its own oracle, we have
$$\Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{O}^c_{priv}(\cdot), \mathcal{P}(\cdot)^{\mathcal{O}^c_{pub}(\cdot)}} = 1\right] \leq \frac{71}{72}.$$

We first consider the case that $\tilde{\mathcal{O}}$ is a real QRO, namely, $\tilde{\mathcal{O}} = \mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{F}_{n,m}$. Note that $\mathcal{D}_q$ will first send a quantum query with state $|\phi\rangle$ to $\tilde{\mathcal{O}}(\cdot) = \mathcal{O}^q(\cdot)$ via the public interface $\mathcal{O}^q_{pub}(\cdot)$ in step 2, and obtain a state

$$|\psi\rangle = \frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\tilde{x}\in X} (-1)^{x^s\cdot\tilde{x}^s} |\tilde{x}, \mathcal{O}^q(\tilde{x})\rangle .$$

If $b = 0$, measuring the state $|\psi\rangle$ will always obtain a pair of $\tilde{x} \in \{0,1\}^n$ and $\tilde{y} \in \{0,1\}^m$ such that $\tilde{x} \in X$ (i.e., $\tilde{x}^{\bar{s}} = x^{\bar{s}}$) and $\tilde{y} = \mathcal{O}^q(\tilde{x})$, which means that $\mathcal{D}_q$ will never output 0 in step 3.1. As for $b = 1$ in step 3.2, $\mathcal{D}_q$ will first send back $|\psi\rangle$ to the oracle $\tilde{\mathcal{O}}(\cdot) = \mathcal{O}^q(\cdot)$ via the public interface $\mathcal{O}^q_{pub}(\cdot)$, and obtain a state

$$|\zeta\rangle = \frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\tilde{x}\in X} (-1)^{x^s\cdot\tilde{x}^s} |\tilde{x}\rangle |0_m\rangle .$$

Then, it will apply $H^s \otimes \mathsf{Id}_m$ on the state $|\zeta\rangle$, and obtain a state $|\zeta'\rangle = |x\rangle|0_m\rangle$. Thus, measuring the state $|\zeta'\rangle$ will result in a pair of $\hat{x} = x$ and $\hat{y} = 0_m$. This shows that $\mathcal{D}_q$ will always output 1 in both cases if $\tilde{\mathcal{O}}$ is a real QRO.

We now consider the case that $\tilde{\mathcal{O}}$ is a simulated QRO, namely, $\tilde{\mathcal{O}} = (\mathcal{O}^c_{priv}(\cdot), \mathcal{P}(\cdot)^{\mathcal{O}^c_{pub}(\cdot)})$. Note that after obtaining the response $|\psi\rangle$ from $\mathcal{P}(\cdot)^{\mathcal{O}^c_{pub}(\cdot)}$ to the query $|\phi\rangle$, the distinguisher $\mathcal{D}(\cdot)$ will first uniformly pick a bit $b \xleftarrow{\$} \{0,1\}$ at random, and then perform different checks depending on the value of b. Let ϑ_0 and ϑ_1 be the probabilities that $|\psi\rangle$ passes the checks for $b = 0$ and $b = 1$, respectively. Then, the probability that $\mathcal{D}(\cdot)$ will output 1 is $(\vartheta_0 + \vartheta_1)/2$.

Note that if $\vartheta_1 \le 35/36$, the proof is already finished as $(\vartheta_0+\vartheta_1)/2 \le 71/72$. Thus, it suffices to consider that $\vartheta_1 > 35/36$. Let $z = \hat{x} \oplus x$ be the difference between the bit string $\hat{x} \in \{0,1\}^n$ obtained in step 3.2 and the random string $x \in \{0,1\}^n$ chosen in step 1. By definition, we have that $\vartheta_1 \le \Pr[z = 0_n]$. We now give a bound on ϑ_0. Let E be the event that $\mathcal{P}$ makes a classical query $\tilde{x} \in \{0,1\}^n$ such that $\tilde{x}^{\bar{s}} = x^{\bar{s}}$. Note that if E does not happen, the probability that $\mathcal{P}$ passes the check in step 3.1 is at most $1/2^m$. Thus, we have $\vartheta_0 \le 1/2^m + \Pr[E]$. Moreover, let $\delta = 1/2 - \nu$ for some $0 < \nu < 1/2$, by the law of total probability we have that $\Pr[E]$

$$\begin{aligned} &= \Pr[E \,|\mathsf{hw}(\bar{s}) \le \delta n\,] \cdot \Pr[\mathsf{hw}(\bar{s}) \le \delta n] + \Pr[E \,|\mathsf{hw}(\bar{s}) > \delta n\,] \cdot \Pr[\mathsf{hw}(\bar{s}) > \delta n] \\ &\le \Pr[\mathsf{hw}(\bar{s}) \le \delta n] + \Pr[E|\mathsf{hw}(\bar{s}) > \delta n]. \end{aligned}$$

Since s is uniformly chosen from $\{0,1\}^n$, we have that $\Pr[\mathsf{hw}(\bar{s}) \le \delta n] \le e^{-n\nu^2}$ by the Chernoff Bounds. Furthermore, let e be the classical information about x that $\mathcal{P}$ obtains from the interactions, and let $I(x,e)$ be the mutual information between x and e. Then, we have that $I(x^{\bar{s}}, e) \le I(x,e)$, and

$$\Pr[E \,|\mathsf{hw}(\bar{s}) > \delta n\,] \le \frac{\ell}{2^{\mathsf{hw}(\bar{s}) - I(x^{\bar{s}},e)}} \le \frac{\ell}{2^{\delta n - I(x,e)}},$$

where $\ell = \mathrm{poly}(\kappa)$ is the number of classical queries made by $\mathcal{P}$ (since $I(x,e)$ bounds the information $\mathcal{P}$ obtained about x from above, see Section B). Thus, we have that

$$\vartheta_0 \le \frac{1}{2^m} + e^{-n\nu^2} + \frac{\ell}{2^{\delta n - I(x,e)}}.$$

Note that if we only care about how much information $\mathcal{P}$ obtains about x from the input state $|\phi\rangle$, we can simplify the analysis by temporarily treating the output register of the query (which is initialized with zeros) as a part of the ancillary register of $\mathcal{P}$ and omit the check of $\hat{y}$. Since the choice of b is random and independent from x and s, the algorithm $\mathcal{P}$ cannot obtain more information about (x,s) when $b = 0$. This means that we can use Lemma 3 to establish a connection between $I(x,e)$ and ϑ_1 (because in this simplified case, the input to $\mathcal{P}$ and the check in $b = 1$ are essentially the same to the that in Lemma 3):

$$I(x,e) \le n(\alpha + \frac{1}{\alpha} \sum_{\mathsf{hw}(z)\ge 1} \Pr[z]) = n(\alpha + \frac{1}{\alpha}(1 - \Pr[z = 0_n])) \le n(\alpha + \frac{1}{\alpha}(1 - \vartheta_1)),$$

where $\alpha > 0$ is an arbitrary real. Since $\vartheta_1 > 35/36$, by setting $\alpha = 1/6$ we have $I(x,e) \leq n(\alpha + (1-\vartheta_1)/\alpha) \leq n/3$. Moreover, by setting $\nu = 1/8$ we have that

$$\vartheta_0 \leq \frac{1}{2^m} + e^{-n\nu^2} + \frac{\ell}{2^{\delta n - I(x,e)}} \leq \frac{1}{2^m} + e^{-n/64} + \frac{\ell}{2^{n/24}} \leq \frac{1}{2} + \mathrm{negl}(\kappa)$$

holds for sufficiently large $n = \omega(\log \kappa)$. Thus, for $\vartheta_1 > 35/36$, we also have

$$\Pr\left[\mathcal{D}(1^\kappa)^{\mathcal{O}_{priv}(\cdot),\mathcal{P}(\cdot)^{\mathcal{O}_{pub}(\cdot)}} = 1\right] = \frac{\vartheta_0 + \vartheta_1}{2} \leq \frac{3}{4} + \mathrm{negl}(\kappa) \leq \frac{71}{72}$$

for sufficiently large $n = \omega(\log \kappa)$. This completes the proof. □

Remark 1 (On the Implications of Theorem 2). First, one can already use a tighter bound $1/68 > 1/72$ for a reasonable choice of $(n, m, \ell) = (256, 128, 2^{32})$. Second, by running the above distinguisher $\mathcal{D}_q$ a polynomial number of times, one can obtain an algorithm $\mathcal{D}'$ which always outputs 1 when it is in the QROM, and outputs 0 with probability negligibly close to 1 when it is in the ROM. Third, by using algorithm $\mathcal{D}'$ as a sub-routine, one can construct an artificial system $\mathcal{C}'$ from another system $\mathcal{C}$ (e.g., a signature scheme) that is secure in the ROM, such that $\mathcal{C}'$ first runs algorithm $\mathcal{D}'$ and reveals some secret information (e.g., a signing key) of $\mathcal{C}$ if $\mathcal{D}'$ outputs 1, otherwise performs normally as $\mathcal{C}$ does if $\mathcal{D}'$ outputs 0 (just like the proof of Theorem 3). Clearly, $\mathcal{C}'$ is secure in the ROM (as $\mathcal{D}'$ will almost always output 0, and $\mathcal{C}$ is secure in the ROM), but it is insecure in the QROM (since $\mathcal{D}'$ will always output 1 in the QROM). Note that this artificial construction is actually implied by Lemma 2. We also note that the system $\mathcal{C}'$ will send out a polynomial number of random quantum queries (as $\mathcal{D}'$ will do), which may be unlikely to happen in a real system. In next section, we will give a pure classical distinguisher by showing the quantum computation of $\mathcal{D}_q$ can be delegated to any distrust quantum algorithm under the LWE assumptions.

3.2 On the Failure of the QROM

It is well-known that there exist schemes that are secure in the ROM, but for which any implementation of the RO results in insecure schemes [12,26,25]. Given that the QROM is stronger than the ROM, one might wonder if the QROM has some essentially different properties such that a security proof in the QROM could somehow guarantee the security of a cryptosytem in the real world. Now, we adapt the techniques in [25] to show that the answer is negative. Formally,

Theorem 3. *If there is an efficient post-quantum signature scheme which is secure in the QROM, then there exists another efficient signature scheme that is also secure in the QROM, but for which any implementation of the QRO using arbitrary efficiently computable function is insecure.*

The key observation behind our proof is that a QRO $\mathcal{O}^q(\cdot) \xleftarrow{\$} \mathcal{F}_{n,m}$ cannot be fully determined by a bit string of length less than $m \cdot 2^n$ (as each n-bit classical input will be mapped into a random m-bit string), but an efficiently computable function always has a succinct representation which can be uniquely encoded into a polynomial number of bits. Thus, any efficient implementation of the QRO actually allows the adversary to get a succinct representation of "the QRO", which is unlikely to happen for an adversary in the QROM.

For our purpose, we restrict our attention to post-quantum signature schemes where the (quantum) adversary is only allowed to make classical signing queries, and do not consider the case that the adversary can make quantum signing queries [9]. Now, we are ready to give the formal proof of Theorem 3.

Proof. Let $\mathcal{SIG} = (\mathsf{KeyGen}^{\mathcal{O}_1(\cdot)}(\cdot), \mathsf{Sign}^{\mathcal{O}_1(\cdot)}(\cdot,\cdot), \mathsf{Verify}^{\mathcal{O}_1(\cdot)}(\cdot,\cdot,\cdot))$ be an efficient post-quantum signature in the QROM, where $\mathcal{O}_1(\cdot) \xleftarrow{\$} \mathcal{F}_{n,m}$ is a QRO. Let κ be the security parameter. We modify $\mathcal{SIG}$ to obtain a new signature scheme $\widetilde{\mathcal{SIG}} = (\mathsf{KeyGen}^{\mathcal{O}_1(\cdot)}(\cdot), \widetilde{\mathsf{Sign}}^{\mathcal{O}_1(\cdot),\mathcal{O}_2(\cdot)}(\cdot,\cdot), \widetilde{\mathsf{Verify}}^{\mathcal{O}_1(\cdot)}(\cdot,\cdot,\cdot))$, which uses another QRO $\mathcal{O}_2(\cdot) \xleftarrow{\$} \mathcal{F}_{\omega(\log\kappa),1}$. Specifically, given the signing key sk and a message $\mu \in \{0,1\}^*$ as inputs, the signing algorithm $\widetilde{\mathsf{Sign}}^{\mathcal{O}_1(\cdot),\mathcal{O}_2(\cdot)}(sk, \mu)$ first computes $b \leftarrow \mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu)$. If $b = 1$, the algorithm $\widetilde{\mathsf{Sign}}^{\mathcal{O}_1(\cdot),\mathcal{O}_2(\cdot)}(sk, \mu)$

directly outputs sk. Otherwise, it runs $\mathsf{Sign}^{\mathcal{O}_1(\cdot)}(sk,\mu)$ and outputs whatever $\mathsf{Sign}^{\mathcal{O}_1(\cdot)}(sk,\mu)$ returns. Accordingly, given the verification key vk, a message $\mu \in \{0,1\}^*$ and a signature σ as inputs, the algorithm $\widetilde{\mathsf{Verify}}^{\mathcal{O}_1(\cdot)}(vk,\mu,\sigma)$ returns 1 if σ is a valid signing key for vk, otherwise returns whatever $\mathsf{Verify}^{\mathcal{O}_1(\cdot)}(vk,\mu,\sigma)$ outputs.

Clearly, the signature scheme $\widetilde{\mathcal{SIG}}$ is correct. Moreover, if the sub-routine $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu)$ always outputs 0 for any message $\mu \in \{0,1\}^*$, then the new scheme $\widetilde{\mathcal{SIG}}$ essentially has the same functionality and security as the original scheme $\mathcal{SIG}$. We now construct a sub-routine $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\cdot)$ with two useful properties: 1) for a real QRO $\mathcal{O}_2(\cdot) \xleftarrow{\$} \mathcal{F}_{\omega(\log\kappa),1}$, the probability that there is a $\mu \in \{0,1\}^*$ such that $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu)$ outputs 1 is negligible; and 2) for any efficient implementation of $\mathcal{O}_2(\cdot)$, it is easy to find a special $\mu^* \in \{0,1\}^*$ such that $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu^*)$ always outputs 1. Formally, $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\cdot)$ works as follows:

1. Given an input $\mu \in \{0,1\}^*$, parse μ as a pair (π,t) consisting of an encoding of a universal turing machine π and a unary encoding of some integer t. Let ℓ_μ and ℓ_π be the length of μ and π, respectively (i.e., $t+\ell_\pi=\ell_\mu$). Then, do the following computation:
   (a) Set $\ell = 2\ell_\pi + \kappa$, and simulate at most t steps of the turing machine π to obtain $\alpha_1 = \pi(1), \cdots, \alpha_\ell = \pi(\ell)$. If the simulation stops before obtaining $\alpha_j = \pi(j)$ for some $j \le \ell$, set $\alpha_j = \alpha_{j+1} = \cdots = \alpha_\ell = 0$.
   (b) Send ℓ queries with $x = 1, \cdots, \ell$ to (the private interface of) the oracle $\mathcal{O}_2(\cdot)$, and obtain the responses $\beta_1 = \mathcal{O}_2(1), \ldots, \beta_\ell = \mathcal{O}_2(\ell)$;
   (c) If $\alpha_i = \beta_i$ for all $i \in \{1,\ldots,\ell\}$, output 1, else output 0.

Obviously, $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\cdot)$ runs in polynomial time. Moreover, for any fixed $\pi \in \{0,1\}^*$, the probability that $\pi(x) = \mathcal{O}_2(x)$ is at most $1/2$ over the random choice of $\mathcal{O}_2(\cdot) \xleftarrow{\$} \mathcal{F}_{\omega(\log\kappa),1}$. Since $\mathcal{D}(\mu)^{\mathcal{O}_2(\cdot)}$ will compare the values of $\mathcal{O}_2(\cdot)$ and $\pi(\cdot)$ at inputs $x \in \{1,\ldots,\ell\}$, for any fixed μ (and thus fixed π) the probability that $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu)$ outputs 1 is at most $1/2^\ell$. Taken over all possible π of a fixed length ℓ_π, the probability that $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu)$ outputs 1 is at most $2^{\ell_\pi}/2^\ell = 2^{-(\ell_\pi+\kappa)}$. Thus, the probability that there exists an $\mu \in \{0,1\}^*$ such that $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu)$ outputs 1 is at most $\sum_{\ell_\pi=1}^{\infty} 2^{-(\ell_\pi+\kappa)} \le 2^{-\kappa}$. Since this probability only depends on the choice of the QRO $\mathcal{O}_2(\cdot) \xleftarrow{\$} \mathcal{F}_{\omega(\log\kappa),1}$ and is independent of the choice of message $\mu \in \{0,1\}^*$, it is infeasible even for an adversary given quantum access to $\mathcal{O}_2(\cdot)$ to output a message $\mu \in \{0,1\}^*$ such that $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu)$ outputs 1 with non-negligible probability. This shows that the signature scheme $\widetilde{\mathcal{SIG}}$ is secure in the QROM.

Now, consider that $\mathcal{O}_2(\cdot)$ is implemented by using a function $f \in \mathcal{F}_{\omega(\log\kappa),1}$ which can be computed in polynomial time. Let π_f be an encoding of a universal turing machine that efficiently computes $f(\cdot)$. Let $\ell' = 2\ell_{\pi_f} + \kappa$, where ℓ_{π_f} is the length of π_f. Let t' be the maximal number of steps that the turing machine $\pi_f(\cdot)$ is required for computing $\pi_f(1),\ldots,\pi_f(\ell')$, and let μ^* be the concatenation of π_f and a unary encoding of t'. Since $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu^*)$ can compute $\pi_f(1),\ldots,\pi_f(\ell')$ in less than t' steps, and $\pi_f(x) = f(x) = \mathcal{O}_2(x)$ holds for any input $x \in \{0,1\}^{\omega(\log\kappa)}$, the sub-routine $\mathcal{D}^{\mathcal{O}_2(\cdot)}(\mu^*) = \mathcal{D}^{f(\cdot)}(\mu^*)$ will always output 1. Thus, for any efficiently computable function $f \in \mathcal{F}_{\omega(\log\kappa),1}$ that implements $\mathcal{O}_2(\cdot)$, there is an adversary which can obtain the signing key sk of $\widetilde{\mathcal{SIG}}$ by making a single signing query. This completes the proof. □

4 Classical Distinguisher with Exponential Query Gaps

At a high level, the quantum distinguisher in Section 3 needs two quantum queries for the following two goals: 1) the first quantum query is used to hide a random pattern that is not known by $\mathcal{P}$; 2) the second quantum query is used to check if $\mathcal{P}$ returns a correct answer to the first quantum query (by either taking a standard measurement or a Hadamard measurement). In this section, we show that the quantum computations (e.g., to generate the two quantum queries) of $\mathcal{D}$ can be delegated to any distrust quantum algorithms in Section 4.2 and Section 4.3 by adapting the results of Brakerski et al. [10] and Mahadev [24].

4.1 (Extended) Noisy Trapdoor Claw-Free Functions

In this subsection, we first recall the definitions of extended noisy trapdoor claw-free functions (NTCF) from [24].

Definition 2 (NTCF Family). *Let λ be a security parameter. Let $\mathcal{X}$ and $\mathcal{Y}$ be a finite sets. Let $\mathcal{K}_{\mathcal{F}}$ be a finite set of keys. A family of functions*

$$\mathcal{F} = \{f_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}}\}_{k \in \mathcal{K}_{\mathcal{F}}, b \in \{0,1\}}$$

is called a noisy trapdoor claw-free (NTCF) family if the following conditions hold:

1. **Efficient Function Generation.** *There exists an efficient probabilistic algorithm $\mathsf{GEN}_{\mathcal{F}}$ which generates a key $k \in \mathcal{K}_{\mathcal{F}}$ together with a trapdoor t_k:*

$$(t, t_k) \leftarrow \mathsf{GEN}_{\mathcal{F}}(1^{\lambda}).$$

2. **Trapdoor Injective Pair.** *For all keys $k \in \mathcal{K}_{\mathcal{F}}$ be the following conditions hold.*
   (a) *Trapdoor: For all $b \in \{0,1\}$ and $x \neq x' \in \mathcal{X}$, $\mathsf{SUPP}(f_{k,b}(x)) \cap \mathsf{SUPP}(f_{k,b}(x')) = \emptyset$. Moreover, there exists an efficient deterministic algorithm $\mathsf{INV}_{\mathcal{F}}$ such that for all $b \in \{0,1\}, x \in \mathcal{X}$ and $y \in \mathsf{SUPP}(f_{k,b}(x)), \mathsf{INV}_{\mathcal{F}}(t_k, b, y) = x$.*
   (b) *Injective pair: There exists a perfect matching $\mathcal{R}_k \subseteq \mathcal{X} \times \mathcal{X}$ such that $f_{k,b}(x_0) = f_{k,b}(x_1)$ if and only if $(x_0, x_1) \in \mathcal{R}_k$.*
3. **Efficient Range Superposition.** *For all keys $k \in \mathcal{K}_{\mathcal{F}}$ and $b \in \{0,1\}$ there exists a function $f'_{k,b:\mathcal{X} \to \mathcal{D}_{\mathcal{Y}}}$ such that*
   (a) *For all $(x_0, x_1) \in \mathcal{R}_k$ and $y \in \mathsf{SUPP}(f'_{k,b}(x_b)), \mathsf{INV}_{\mathcal{F}}(t_k, b, y) = x$ and $\mathsf{INV}_{\mathcal{F}}(t_k, b\oplus, y) = x_{b\oplus 1}$.*
   (b) *There exists an efficient deterministic procedure $\mathsf{CHK}_{\mathcal{F}}$ that, on input $k, b \in \{0,1\}, x \in \mathcal{X}$ and $y \in \mathcal{Y}$, returns 1 if $y \in \mathsf{SUPP}(f'_{k,b}(x))$ and 0 otherwise.*
   (c) *For every k and $b \in \{0,1\}$,*

$$E_{x \xleftarrow{\$} \mathcal{X}}[H^2(f_{k,b}(x), f'_{k,b}(x))] \leq \mu(\lambda),$$

   for some negligible function $\mu(\cdot)$. Here H^2 is the Hellinger distance. Moreover, there exists an efficient procedure $\mathsf{SAMP}_{\mathcal{F}}$ that on input k and $b \in \{0,1\}$ prepares the state

$$\frac{1}{\sqrt{|X|}\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \sqrt{(f'_{k,b}(x))(y)}\,|x\rangle\,|y\rangle}.$$

4. **Adaptive Hardcore Bit.** *For all keys $k \in \mathcal{K}_{\mathcal{F}}$ the following conditions hold, for some integer w that is a polynomially bounded function of λ.*
   (a) *For all $b \in \{0,1\}$ and $x \in \mathcal{X}$, there exists a set $G_{k,b,x} \subseteq \{0,1\}^w$ such that $\Pr_{d \xleftarrow{\$} \{0,1\}^w}[d \notin G_{k,b,x}]$ is negligible, and moreover there exists an efficient algorithm that checks for membership in $G_{k,b,x}$ given k, b, x and the trapdoor t_k.*
   (b) *There is an efficiently computable injection $J : \mathcal{X} \to \{0,1\}^w$, such that J can be inverted efficiently on its range, and such that the following holds. If*

$$\begin{aligned} H_k = \{(b, x_b, d, d \cdot (J(x_0) \oplus J(x_1)))|b \in \{0,1\}, (x_0, x_1) \in \mathcal{R}_k, \\ d \in G_{k,0,x_0} \cap G_{k,1,x_1}\}, \\ \bar{H}_k = \{(b, x_b, d, c)|(b, x, d, \oplus 1) \in H_k\}, \end{aligned}$$

   then for any quantum polynomial time procedure $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that

$$\left| \Pr_{(k,t_k) \xleftarrow{\$} \mathsf{GEN}_{\mathcal{F}}(1^{\lambda})}[\mathcal{A}(k) \in H_k] - \Pr_{(k,t_k) \xleftarrow{\$} \mathsf{GEN}_{\mathcal{F}}(1^{\lambda})}[\mathcal{A}(k) \in \bar{H}_k] \right| \leq \mu(\lambda).$$

Definition 3 (Trapdoor Injective Function Family). *Let λ be a security parameter. Let $\mathcal{X}$ and $\mathcal{Y}$ be a finite sets. Let $\mathcal{K}_{\mathcal{G}}$ be a finite set of keys. A family of functions*

$$\mathcal{G} = \{g_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}}\}_{k \in \mathcal{K}_{\mathcal{G}}, b \in \{0,1\}}$$

is called a trapdoor injective family if the following conditions hold:

1. **Efficient Function Generation.** *There exists an efficient probabilistic algorithm* $\mathsf{GEN}_{\mathcal{G}}$ *which generates a key* $k \in \mathcal{K}_{\mathcal{G}}$ *together with a trapdoor* t_k*:*

$$(t, t_k) \leftarrow \mathsf{GEN}_{\mathcal{G}}(1^\lambda).$$

2. **Disjoint Trapdoor Injective Pair.** *For all keys* $k \in \mathcal{K}_{\mathcal{G}}$*, for all* $b, b' \in \{0,1\}$ *and* $x, x' \in \mathcal{X}$*, if* $(b, x) \neq (b', x')$*,* $\mathsf{SUPP}(g_{k,b}(x)) \cap \mathsf{SUPP}(g_{k,b'}(x')) = \emptyset$*. Moreover, there exists an efficient deterministic algorithm* $\mathsf{INV}_{\mathcal{G}}$ *such that for all* $b \in \{0,1\}, x \in \mathcal{X}$ *and* $y \in \mathsf{SUPP}(g_{k,b}(x))$*,* $\mathsf{INV}_{\mathcal{G}}(t_k, y) = (b, x)$*.*
3. **Efficient Range Superposition.** *For all keys* $k \in \mathcal{K}_{\mathcal{F}}$ *and* $b \in \{0,1\}$
   (a) *There exists an efficient deterministic procedure* $\mathsf{CHK}_{\mathcal{G}}$ *that, on input* $k, b \in \{0,1\}, x \in \mathcal{X}$ *and* $y \in \mathcal{Y}$*, returns* 1 *if* $y \in \mathsf{SUPP}(g_{k,b}(x))$ *and 0 otherwise.*
   (b) *There exists an efficient procedure* $\mathsf{SAMP}_{\mathcal{G}}$ *that on input* k *and* $b \in \{0,1\}$ *prepares the state*

$$\frac{1}{\sqrt{|X|}\sum_{x\in\mathcal{X}, y\in\mathcal{Y}}}\sqrt{(g_{k,b}(x))(y)}\,|x\rangle\,|y\rangle\,.$$

Definition 4 (Injective Invariance). *A noisy trapdoor claw-free family* $\mathcal{F}$ *is injective invariant if there exists a trapdoor injective family* $\mathcal{G}$ *such that:*

1. *The algorithms* $\mathsf{CHK}_{\mathcal{F}}$ *and* $\mathsf{SAMP}_{\mathcal{F}}$ *are the same as the algorithms* $\mathsf{CHK}_{\mathcal{G}}$ *and* $\mathsf{SAMP}_{\mathcal{G}}$*;*
2. *For all quantum polynomial-time procedures* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *such that*

$$\left|\Pr_{(k,t_k)\xleftarrow{\$}\mathsf{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k)=0] - \Pr_{(k,t_k)\xleftarrow{\$}\mathsf{GEN}_{\mathcal{G}}(1^\lambda)}[\mathcal{A}(k)=0]\right| \leq \mu(\lambda).$$

Definition 5 (Extended Trapdoor Claw-Free Family). *A noisy trapdoor claw-free family* $\mathcal{F}$ *is an extended trapdoor claw-free family if:*

1. *It is injective invariant.*
2. *For all* $k \in \mathcal{K}_{\mathcal{F}}$ *and* $d \in \{0,1\}^w$*, let*

$$H'_{k,d} = \{d \cdot (J(x_0) \oplus J(x_1)) | (x_0, x_1) \in \mathcal{R}_k\}.$$

For all quantum polynomial time procedure $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *and a string* d *such that*

$$\left|\Pr_{(k,t_k)\xleftarrow{\$}\mathsf{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k) \in H'_{k,d}] - \frac{1}{2}\right| \leq \mu(\lambda).$$

As shown in Brakerski et al. [10] and Mahadev [24], one can construct extended NTCF family under the LWE assumptions. We omit the details.

4.2 Generating a Query with a Random Pattern in Superposition

In this section, we give a protocol between a classical verifier $\mathcal{V}$ and a quantum device $\mathcal{P}$ such that $\mathcal{V}$ can be ensured that $\mathcal{P}$ holds a (not necessarily pure) state

$$|x\rangle_s = H^s\,|x\rangle = \frac{1}{\sqrt{2^{\mathsf{hw}(s)}}}\sum_{\tilde{x}\in X}(-1)^{x^s\cdot\tilde{x}^s}\,|\tilde{x}\rangle\,.$$

Let λ be a security parameter, $N \geq 1$ an integer, and $\gamma, q > 0$ functions of λ and n. Let $\mathcal{F}$ be an extended NTCF family, and $\mathcal{G}$ be its corresponding trapdoor injective family $\mathcal{G}$. Now, we will give a protocol between a classical verifier $\mathcal{V}$ and a quantum algorithm $\mathcal{P}$ such that $\mathcal{V}$ can be ensured that $\mathcal{P}$ has indeed generated a (not necessarily pure) state

$$|r \wedge \bar{s}\rangle_s = H^s\,|r \wedge \bar{s}\rangle = \frac{1}{\sqrt{2^{\mathsf{hw}(s)}}}\sum_{\tilde{r}\in R}|\tilde{r}\rangle$$

for some $r, s \in \{0,1\}^n$, and $r^{\bar{s}} \in \{0,1\}^{|\bar{s}|}$ has high mini-entropy even conditioned all the other information holding by $\mathcal{P}$, where $R = \{\tilde{r} \in \{0,1\}^n : \tilde{r}^{\bar{s}} = r^{\bar{s}}\}$ determined by $r, s \in \{0,1\}^n$ (note that $(r \wedge \bar{s})^{\bar{s}} = r^{\bar{s}}$ by definition). Mentioning ahead, the state $|r \wedge \bar{s}\rangle_s$ is exactly the quantum RO query that we want $\mathcal{P}$ to generate in an obvious way. For convenience, we describe the protocol in two phases: the generation phase and the verification phase as in Fig. 3.

Protocol 1

Inputs: The classical algorithm $\mathcal{V}$ inputs an integer n and a real $q \in (0,1)$. The quantum algorithm $\mathcal{P}$ inputs an integer n.

Generation: This is to generate a quantum state holding by $\mathcal{P}$.

1. For $i = 1, \ldots, n$, $\mathcal{V}$ selects $s_i \in \{0,1\}$ such that $\Pr[s_i = 1] = q$. Then, it generates $(k_i, t_i) \leftarrow \mathsf{GEN}_{\mathcal{F}}(1^\lambda)$ if $s_i = 1$, otherwise $(k_i, t_i) \leftarrow \mathsf{GEN}_{\mathcal{G}}(1^\lambda)$. Send the keys $K = (k_1, \ldots, k_n)$ to $\mathcal{P}$.
2. $\mathcal{P}$ first applies the Hadamard operation to n qubits containing 0 to obtain a state

$$\frac{1}{2^{n/2}} \sum_{r=r_1 \ldots r_n \in \{0,1\}^n} |r\rangle = \frac{1}{2^{n/2}} \sum_{r=r_1 \ldots r_n \in \{0,1\}^n} |r_1\rangle \ldots |r_n\rangle .$$

Then, for $i = 1, \ldots, n$, it applies the $\mathsf{SAMP}_{\mathcal{F}} = \mathsf{SAMP}_{\mathcal{G}}$ procedure in superposition with k_i and the i-th qubit containing r_i as input. Let $g'_{k_i,b} = f'_{k_i,b}$ if $s_i = 1$, otherwise $g'_{k_i,b} = g_{k_i,b}$. $\mathcal{P}$ will obtain a state

$$\left|\phi^{(1)}\right\rangle = \frac{1}{2^{n/2}|\mathcal{X}|^{n/2}} \sum_{\substack{r = r_1 \ldots r_n \in \{0,1\}^n, \\ X = x_1 \ldots x_n \in \mathcal{X}^n, \\ Y = y_1 \ldots y_n \in \mathcal{Y}^n}} \alpha_{X,Y} |r\rangle |X\rangle |Y\rangle ,$$

where $\alpha_{X,Y} = \prod_i \sqrt{g'_{k_i,r_i}(x_i)(y_i)}$. Finally, $\mathcal{P}$ measures the registers containing Y to obtain a set $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_n) \in \mathcal{Y}^n$ and a state $\left|\phi^{(2)}\right\rangle$. Send $\hat{Y}$ to $\mathcal{V}$.

Verification: This is to verify the quantum state $|\phi\rangle$ holding by $\mathcal{P}$.

3. For $i = 1, \ldots, n$:
   - 3.1 $\mathcal{V}$ randomly chooses $c_i \xleftarrow{\$} \{0,1\}$ if $s_i = 1$, otherwise set $c_i = 1$. Then, it sends c_i to $\mathcal{P}$.
   - 3.2 $\mathcal{P}$ performs the following computation depending on the value of c_i:
     - (a) In case $c_i = 0$, evaluate the function J on the qubits containing x_i, and then apply a Hadamard transform to the $w+1$ qubits containing r_i and x_i. Measure the $w+1$ registers to obtain a pair $(u_i, d_i) \in \{0,1\} \times \{0,1\}^w$. Send $(\hat{u}_i, \hat{d}_i)$ to $\mathcal{V}$;
     - (b) In case $c_i = 1$, measure the $w+1$ qubits containing (r_i, x_i) to obtain a pair $(\hat{r}_i, \hat{x}_i) \in \{0,1\} \times \mathcal{X}$. Send $(\hat{r}_i, \hat{x}_i)$ to $\mathcal{V}$.
   - 3.3 $\mathcal{V}$ performs the following computation depending on the value of c_i:
     - (a) In case $c_i = 0$, randomly choose $W_i \xleftarrow{\$} \{0,1\}$ if $\hat{d} \notin \hat{G}_{y_i}$. Otherwise, set $W_i = 1$ if $\hat{d}_i \cdot (J(\hat{x}_{0,\hat{y}_i}) \oplus J(\hat{x}_{1,\hat{y}_i})) = \hat{u}_i$, and $W_i = 0$ if not, where $\hat{x}_{b,\hat{y}_i} = \mathsf{INV}_{\mathcal{F}}(t_i, b, \hat{y}_i)$ for $b \in \{0,1\}$.
     - (b) In case $c_i = 1$: set W_i as the value returned by $\mathsf{CHK}_{\mathcal{G}}(k, b, x, y)$.

Outputs: $\mathcal{V}$ outputs 0 if $\sum_{i:s_i=1} W_i < (1-\gamma)qn$, and 1 otherwise.

Fig. 3. Generating a Query with a Random Pattern in Superposition

Now, we have the following results.

Theorem 4. *If $\mathcal{F}$ is an extended NTCF family, and $\mathcal{G}$ is the corresponding trapdoor injective family. Let $s = s_1 \ldots s_n \in \{0,1\}$ be the secret bit string chosen by $\mathcal{V}$, and $\hat{Y} = (\hat{y}_1, \ldots, \hat{y}_n) \in \mathcal{Y}^n$ be the set output by $\mathcal{P}$ in the generation phase, respectively. Moreover, let $\hat{x}_{r_i,y_i} = \mathsf{INV}_{\mathcal{F}}(t_i, r_i, \hat{y}_i)$ for any $r_i \in \{0,1\}$ if $s_i = 1$, otherwise $(\hat{r}_i, \hat{x}_{r_i,\hat{y}_i}) = \mathsf{INV}_{\mathcal{G}}(t_i, \hat{y}_i)$. Then, in an honest execution of Protocol 1 in Fig. 3 between a classical*

verifier $\mathcal{V}$ and a quantum algorithm $\mathcal{P}$, the quantum state $\left|\phi^{(2)}\right\rangle$ obtained by $\mathcal{P}$ in the generation phase is within negligible trace distance from the following state:

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r=r_1\dots r_n \in R_{s,\hat{Y}},\\ \hat{X}_{r,\hat{Y}}=\hat{x}_{r_1,\hat{y}_1}\dots\hat{x}_{r_n,\hat{y}_n}\in\mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle,$$

where $R_{s,\hat{Y}} = \{r = r_1 \dots r_n \in \{0,1\}^n : s_i = 0 \wedge r_i = \hat{r}_i\}$. Moreover, the probability that $\mathcal{V}$ outputs 1 is negligibly close to 1.

Proof. By definition, the quantum state $\left|\phi^{(2)}\right\rangle$ is obtained by directly measuring the qubits of the state $\left|\phi^{(1)}\right\rangle$ containing Y. We first show that $\left|\phi^{(1)}\right\rangle$ is within negligible trace distance from the following state:

$$|\psi\rangle = \frac{1}{2^{n/2}|\mathcal{X}|^{n/2}} \sum_{\substack{r=r_1\dots r_n\in\{0,1\}^n,\\ X=x_1\dots x_n\in\mathcal{X}^n,\\ Y=y_1\dots y_n\in\mathcal{Y}^n}} \alpha'_{X,Y}\,|r\rangle\,|X\rangle\,|Y\rangle\,,$$

where $\alpha'_{X,Y} = \prod_i \sqrt{\hat{g}_{k_i,r_i}(x_i)(y_i)}$, $\hat{g}_{k_i,b} = f_{k_i,b}$ if $s_i = 1$, otherwise $\hat{g}_{k_i,b} = g_{k_i,b}$. Clearly, $\alpha'_{X,Y}$ only differs from $\alpha_{X,Y}$ at the position $s_i = 1$. Now, we define a set of state $\left|\phi_0^{(1)}\right\rangle, \dots, \left|\phi_n^{(1)}\right\rangle$ such that $\left|\phi_0^{(1)}\right\rangle = \left|\phi^{(1)}\right\rangle, \left|\phi_n^{(1)}\right\rangle = |\psi\rangle$, and for $i = 1, \dots, n$ the state $\left|\phi_i^{(1)}\right\rangle$ is obtained from $\left|\phi_{i-1}^{(1)}\right\rangle$ by replacing $g'_{k_i,b}$ with $\hat{g}_{k_i,b}$. As n is a polynomial of λ, in order to show that the trace distance between $\left|\phi^{(1)}\right\rangle$ and $|\psi\rangle$ is negligible, it suffices to show that for any $i > 1$, the trace distance between $\left|\phi_i^{(1)}\right\rangle$ and $\left|\phi_{i-1}^{(1)}\right\rangle$ is negligible in λ, namely,

$$\left\|\left|\phi_i^{(1)}\right\rangle - \left|\phi_{i-1}^{(1)}\right\rangle\right\|_{tr} \le \mathrm{negl}(\lambda).$$

This obviously holds for $s_i = 0$, because in this case $g'_{k_i,b} = \hat{g}_{k_i,b}$ and $\left|\phi_i^{(1)}\right\rangle = \left|\phi_{i-1}^{(1)}\right\rangle$. Thus, we only have to consider the case $s_i = 1$, where $g'_{k_i,b} = f'_{k_i,b}$ and $\hat{g}_{k_i,b} = f_{k_i,b}$. As for every k and $b \in \{0,1\}$, $E_{x\xleftarrow{\$}\mathcal{X}}[H^2(f_{k,b}(x), f'_{k,b}(x))] \le \mu(\lambda)$ for some negligible function $\mu(\cdot)$ by the assumption, we have that

$$\left\|\left|\phi_i^{(1)}\right\rangle - \left|\phi_{i-1}^{(1)}\right\rangle\right\|_{tr} \le \mathrm{negl}(\lambda).$$

By the property of trace distance, we have that $\left|\phi^{(2)}\right\rangle$ is within negligible trace distance from the state by measuring the state

$$|\psi\rangle = \frac{1}{2^{n/2}|\mathcal{X}|^{n/2}} \sum_{\substack{r=r_1\dots r_n\in\{0,1\}^n,\\ X=x_1\dots x_n\in\mathcal{X}^n,\\ Y=y_1\dots y_n\in\mathcal{Y}^n}} \alpha'_{X,Y}\,|r\rangle\,|X\rangle\,|Y\rangle\,.$$

to obtain $\hat{Y} = \{\hat{y}_1, \dots, \hat{y}_n\}$. By the injective pair property of $\mathcal{F}$ and the disjoint trapdoor injective pair property of $\mathcal{G}$, we have that $\left|\phi^{(2)}\right\rangle$ is within negligible trace distance from the state by measuring the state

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r=r_1\dots r_n \in R_{s,\hat{Y}},\\ \hat{X}_{r,\hat{Y}}=\hat{x}_{r_1,\hat{y}_1}\dots\hat{x}_{r_n,\hat{y}_n}\in\mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle.$$

This completes the proof of the first claim.

As for the second claim, by the Chernoff bound it suffices to show that $W_i = 1$ almost always holds for $s_i = 1$. By the definition of an extended NTCF, $\mathcal{D}$ will set $W_i = 0$ for an honest $\mathcal{P}$ only if $\mathcal{P}$ outputs a $d_i \notin \hat{G}_{\hat{y}_i}$, which by item 4(a) in Definition 2 happens with negligible probability. This completes the proof. □

Theorem 5. *If $\mathcal{F}$ is an extended NTCF family, and $\mathcal{G}$ is the corresponding trapdoor injective family. Let $s = s_1 \dots s_n \in \{0,1\}$ be the secret bit string chosen by $\mathcal{V}$, and $\hat{Y} = (\hat{y}_1, \dots, \hat{y}_n) \in \mathcal{Y}^n$ be the set output by $\mathcal{P}$ in the generation phase, respectively. Let $\hat{r} \in \{0,1\}^{\mathsf{hw}(\bar{s})}$ be the string obtained by concatenating each bit $\hat{r}_i \in \{0,1\}$ output by $\mathcal{P}$ satisfying $s_i = 0$ in the verification phase, and let $R_{s,\hat{Y}} = \{r = r_1 \dots r_n \in \{0,1\}^n : r^{\bar{s}} = \hat{r}\}$. Let E be the event that $\mathcal{P}$ outputs a $r \in R_{s,\hat{Y}}$ at the time before the verification phase. Let A be the event that $\mathcal{V}$ outputs 1 in the verification phase. Then, for any quantum polynomial time algorithm $\mathcal{P}$, we have that $\Pr[E|A] \leq \mathrm{negl}(\lambda)$.*

Proof. We first consider a modification of protocol 1 where $\mathcal{V}$ always generates the keys by running $(k_i, t_i) = \mathsf{GEN}_{\mathcal{F}}(1^\lambda)$ even for $s_i = 0$. Then, we can similarly define events E' and A' the same as E and A for the modified protocol. By the assumption that no polynomial time quantum algorithm can distinguish the keys generated by $\mathsf{GEN}_{\mathcal{F}}(1^\lambda)$ from that generated by $\mathsf{GEN}_{\mathcal{G}}(1^\lambda)$. We have the $|\Pr[E'|A'] - \Pr[E|A]| \leq \mathrm{negl}(\lambda)$ for any polynomial time quantum adversary $\mathcal{P}$.

Thus, it suffices to show that $\Pr[E'|A'] \leq \mathrm{negl}(\lambda)$. We note that the above modified protocol is essentially identical to the randomness expansion protocol given in [10] except the following two differences:

1. The verifier in the randomness expansion protocol [10] will only generate a fresh k_i if $i = 1$ or $s_{i-1} = 1$, and will reuse the same key $k_i = k_{i+1} = \cdots = k_j$ for any $i < j$ satisfying $s_i = \cdots = s_{j-1} = 0$ and $s_j = 1$. This is necessary for saving the random bits to generate the keys, because the length of the output random bits of a meaningful randomness expansion protocol must be longer than that of the input ones (which is not required in our modified protocol);
2. At the end of the randomness expansion protocol [10], the verifier will either abort if $\sum_{i:s_i=1} W_i < (1-\gamma)qN$, or output a string $\hat{r} \in \{0,1\}^{\mathsf{hw}(\bar{s})}$ obtained concatenating $r_i \in \{0,1\}$ satisfying $s_i = 0$.

Clearly, the output of $\mathcal{D}$ will not affect the view of $\mathcal{P}$, and the way of using fresh k_i for all $i \in \{1, \dots, n\}$ will not give more advantage to $\mathcal{P}$. In fact, by almost the same proof of [10, Proposition 8.9] for the randomness expansion protocol, we can show that for any quantum polynomial time $\mathcal{P}$, there is a negligible function $\epsilon(\lambda)$ such that if $\mathcal{V}$ outputs 1, then the ϵ-smooth min-entropy of $\hat{r}$ conditioned on all the information obtained by $\mathcal{P}$ at the time before the verification phase (and even the challenges $c = c_1 \dots c_n$ in the verification phase) is at least $O(n)$ under appropriate choices of parameters. We refer the details to [10]. Thus, for the modified protocol, the probability that $\mathcal{P}$ outputs a string $r \in R_{s,\hat{Y}}$ at the time before the verification phase, conditioned on the event that $\mathcal{V}$ outputs 1, is negligible under appropriate choices of parameters, namely, $\Pr[E'|A'] \leq \mathrm{negl}(\kappa)$. This completes the proof. □

4.3 Measuring a State in an Oblivious way

Suppose that there is a classical verifier $\mathcal{V}$ holding a secret bit $\delta \in \{0,1\}$ and a quantum device $\mathcal{P}$ holding an arbitrary state $|\phi\rangle$ (which does not necessarily generated by $\mathcal{P}$). Now, we give a protocol between $\mathcal{V}$ and $\mathcal{P}$ such that depending on the value of δ, $\mathcal{V}$ either obtains the measurement outcome of $|\phi\rangle$ or holds some information that can help $\mathcal{P}$ to compute a state that with trace distance negligibly close to the input state $|\phi\rangle$, without leaking the information of δ to $\mathcal{P}$ (i.e., $\mathcal{P}$ does not know which case it is for $\mathcal{V}$). Let λ be the security parameter λ. Let $\mathcal{F}$ be an extended NTCF family $\mathcal{F}$ associated with a corresponding trapdoor injective family $\mathcal{G}$. The full description of the protocol is given in Fig. 4, which is inspired by the measurement protocol given in [24].

Now, we prove the following results about the protocol.

Theorem 6. *In an honest execution of Protocol 2 in Fig. 4 between a classical verifier $\mathcal{V}$ with input δ and a quantum algorithm $\mathcal{P}$ with input a state $|\phi\rangle = \sum_{r=r_1 \dots r_n \in \{0,1\}^n} \beta_r |r_1, \dots, r_n\rangle$, we have*

- *In case $\delta = 0$, the bit string $\hat{r}$ output by $\mathcal{V}$ is the outcome of a standard measurement on $|\phi\rangle$;*
- *In case $\delta = 1$, there is an efficient quantum algorithm $\mathcal{R}$ which is given $KT = \{(k_i, t_i)\}$ and $|\phi'\rangle$ output by $\mathcal{V}$ and $\mathcal{P}$, computes a state $|\psi\rangle$ within negligible trace distance of $|\phi\rangle$.*

Protocol 2

Inputs: The classical algorithm $\mathcal{V}$ inputs an integer n and a bit $\delta \in (0,1)$. The quantum algorithm $\mathcal{P}$ inputs a quantum state $|\phi\rangle = \sum_{r=r_1\ldots r_n\in\{0,1\}^n} \beta_r |r_1,\ldots,r_n\rangle$.

Description: This is the description of the oblivious measurement protocol.

1. For $i=1,\ldots,n$, $\mathcal{V}$ generates $(k_i,t_i) \leftarrow \mathsf{GEN}_{\mathcal{G}}(1^\lambda)$ if $\delta=0$, otherwise $(k_i,t_i)\leftarrow \mathsf{GEN}_{\mathcal{F}}(1^\lambda)$. Then, send K to $\mathcal{P}$.
2. For$i=1,\ldots,n$, $\mathcal{P}$ first applies the $\mathsf{SAMP}_{\mathcal{F}}=\mathsf{SAMP}_{\mathcal{G}}$ procedure in superposition with k_i and the i-th qubit containing r_i as input. Let $g'_{k_i,b}=g_{k_i,b}$ if $\delta=0$, otherwise $g'_{k_i,b}=f'_{k_i,b}$. This will lead to the following state
$$\left|\phi^{(1)}\right\rangle = \frac{1}{|\mathcal{X}|^{n/2}} \sum_{\substack{r=r_1\ldots r_n\in\{0,1\}^n,\\ X=x_1\ldots x_n\in\mathcal{X}^n,\\ Y=y_1\ldots y_n\in\mathcal{Y}^n}} \alpha_{X,Y}|r\rangle|X\rangle|Y\rangle,$$
where $\alpha_{X,Y}=\beta_r\prod_i\sqrt{g'_{k_i,r_i}(x_i)(y_i)}$. Then, $\mathcal{P}$ measures the $\mathcal{Y}$ registers to obtain a set $\hat{Y}=(\hat{y}_1,\ldots,\hat{y}_n)\in\mathcal{Y}^n$ and a state $|\phi'\rangle$. Send $\hat{Y}$ to $\mathcal{V}$.
3. $\mathcal{V}$ computes $(\hat{r}_i,\hat{x}_{\hat{r}_i,y_i})=\mathsf{INV}_{\mathcal{G}}(t_i,\hat{y}_i)$ for each $\hat{y}_i\in\mathcal{X}$ if $\delta=0$.

Outputs: $\mathcal{V}$ outputs $r=r_1\ldots r_n$ if $\delta=0$, otherwise $KT=\{(k_i,t_i)\}$; $\mathcal{P}$ outputs a state $|\phi'\rangle$

Fig. 4. An Oblivious Measurement Protocol

Proof. As the protocol basically applies to same operations to each qubit of $|\phi\rangle$ independently. It suffices to consider first qubit of $|\phi\rangle$. Without loss of generality, we can rewrite $|\phi\rangle=\sum_{r_1\in\{0,1\}}\hat{\beta}_{r_1}|r_1\rangle|\phi_{r_1}\rangle$. Applying the $\mathsf{SAMP}_{\mathcal{F}}=\mathsf{SAMP}_{\mathcal{G}}$ procedure in superposition with k_1 and the first qubit containing r_1 as input will lead to a state either
$$\frac{1}{\sqrt{|\mathcal{X}|}}\sum_{\substack{r_1\in\{0,1\},\\ x\in\mathcal{X},y\in\mathcal{Y}}}\hat{\beta}_{r_1}\sqrt{g_{k_1,r_1}(x_1)(y_1)}\,|r_1\rangle|\phi_{r_1}\rangle|x_1\rangle|y_1\rangle$$
for $\delta=0$, or
$$\frac{1}{\sqrt{|\mathcal{X}|}}\sum_{\substack{r_1\in\{0,1\},\\ x\in\mathcal{X},y\in\mathcal{Y}}}\hat{\beta}_{r_1}\sqrt{f'_{k_1,r_1}(x_1)(y_1)}\,|r_1\rangle|\phi_{r_1}\rangle|x_1\rangle|y_1\rangle,$$
which then is within negligible trace distance of the following state:
$$\frac{1}{\sqrt{|\mathcal{X}|}}\sum_{\substack{r_1\in\{0,1\},\\ x\in\mathcal{X},y\in\mathcal{Y}}}\hat{\beta}_{r_1}\sqrt{f_{k_1,r_1}(x_1)(y_1)}\,|r_1\rangle|\phi_{r_1}\rangle|x_1\rangle|y_1\rangle,$$
because $E_{x\xleftarrow{\$}\mathcal{X}}[H^2(f_{k,b}(x),f'_{k,b}(x))]\leq\mu(\lambda)$ for some negligible function $\mu(\cdot)$ by the assumption. Measuring the qubits containing y_1 will obtain $\hat{y}_1\in\mathcal{Y}$. Let $(\hat{r}_1,x_{r_1,\hat{y}_1})=\mathsf{INV}_{\mathcal{G}}(t_1,y_1)$ if $\delta=0$, otherwise $\hat{x}_{r_1,\hat{y}_1}=\mathsf{INV}_{\mathcal{F}}(t_1,b,\hat{y}_1)$ for $r_1\in\{0,1\}$. If $\delta=0$, the remaining state $|\phi'\rangle$ holding by $\mathcal{P}$ is
$$|\hat{r}_1\rangle|\phi_{\hat{r}_1}\rangle|x_{\hat{r}_1,\hat{y}_1}\rangle$$
by the disjoint trapdoor injective pair property of $\mathcal{G}$. Otherwise, the remaining state $|\phi'\rangle$ is within negligible trace distance of the following state:
$$\sum_{r_1\in\{0,1\}}\hat{\beta}_{r_1}|r_1\rangle|\phi_{r_1}\rangle|\hat{x}_{r_1,\hat{y}_1}\rangle$$

by the trapdoor and injective pair property of $\mathcal{F}$. Clearly, if $\delta = 0$, $\hat{r}_1$ output by $\mathcal{V}$ is the outcome of a standard measurement on the first qubit of $|\phi\rangle$. Otherwise, given t_1 and $|\phi'\rangle$ as inputs, $\mathcal{P}$ can uncompute the register containing $\hat{x}_{r_1,\hat{y}_1}$ to obtain a state within negligible trace distance of $|\phi\rangle$ by computing $\hat{x}_{r_1,\hat{y}_1} = \mathsf{INV}_{\mathcal{F}}(t_1, r_1, \hat{y}_1)$. This completes the proof of the lemma. □

Theorem 7. *If $\mathcal{F}$ is an extended NTCF family, and $\mathcal{G}$ is the corresponding trapdoor injective family $\mathcal{G}$, then for a uniformly random $\delta \in \{0,1\}$ holding by $\mathcal{V}$ and any polynomial time quantum algorithm $\mathcal{P}$, the probability that $\mathcal{P}$ outputs $\delta' = \delta$ is negligibly close to $1/2$ after interacting with $\mathcal{V}$.*

Proof. Consider a modification of protocol 2 where $\mathcal{V}$ always generates the keys by running $(k_i, t_i) = \mathsf{GEN}_{\mathcal{F}}(1^\lambda)$ even for $s_i = 0$. Then, $\mathcal{P}$ cannot obtain any information of δ by interacting $\mathcal{V}$ in the modified protocol. Thus, the probability that $\mathcal{P}$ output $\delta' = \delta$ is exactly $1/2$. By the fact that $\mathsf{SAMP}_{\mathcal{F}} = \mathsf{SAMP}_{\mathcal{G}}$, and that no polynomial time quantum algorithm $\mathcal{P}$ can distinguish the keys generated by $\mathsf{GEN}_{\mathcal{F}}(1^\lambda)$ from that generated by $\mathsf{GEN}_{\mathcal{G}}(1^\lambda)$, we have the view of $\mathcal{P}$ in the modified protocol and the real protocol 2 are computationally indistinguishable. This means that the probability that $\mathcal{P}$ outputs $\delta' = \delta$ is negligibly close to $1/2$, which completes the proof. □

4.4 A Classical Proof of Quantum Access to a RO

Let $\mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a QRO for sufficient large $n = \mathrm{poly}(\lambda)$ and $m \geq 1$. We now give a protocol between a classical distinguisher $\mathcal{D}^{\mathcal{O}^q_{priv}(\cdot)}$ and a quantum algorithm $\mathcal{P}^{\mathcal{O}^q_{pub}(\cdot)}$ such that $\mathcal{D}^{\mathcal{O}^q_{priv}(\cdot)}$ outputs 1 with probability negligibly close to 1, but for any polynomial time quantum algorithm $\tilde{\mathcal{P}}^{\mathcal{O}^q_{pub}(\cdot)}$, the probability that $\mathcal{D}^{\mathcal{O}^q_{priv}(\cdot)}$ outputs 1 is at most $3/4 + \mathrm{negl}(\kappa)$. Let λ be the security parameter λ. Let $\mathcal{F}$ be an extended NTCF family $\mathcal{F}$ associated with a corresponding trapdoor injective family $\mathcal{G}$. The full description of the protocol is given in Fig. 5.

For our purpose, it suffices to prove the following results.

Theorem 8. *Let $\mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a QRO for some integers $n = \mathrm{poly}(\lambda)$ and $m \geq 1$. If $\mathcal{F}$ is an extended NTCF family, and $\mathcal{G}$ is the corresponding trapdoor injective family. Then, in an honest execution of Protocol 3 in Fig. 5 between a classical distinguisher $\mathcal{D}^{\mathcal{O}^q_{priv}(\cdot)}$ and a quantum algorithm $\mathcal{P}^{\mathcal{O}^q_{pub}(\cdot)}$, the probability that $\mathcal{D}^{\mathcal{O}^q_{priv}(\cdot)}$ outputs 1 is negligibly close to 1.*

Proof. It suffices to show that $\mathcal{D}^{\mathcal{O}^q_{priv}(\cdot)}$ almost always outputs 1 with probability negligibly close to 1 for both $\delta = 0$ and $\delta = 1$. Note that before executing Protocol 2, $\mathcal{P}$ will obtain a state within negligible trace distance from the following state.

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r = r_1 \ldots r_n \in R_{s,\hat{Y}}, \\ \hat{X}_{r,\hat{Y}} = \hat{x}_{r_1,\hat{y}_1} \ldots \hat{x}_{r_n,\hat{y}_n} \in \mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle |\mathcal{O}(r)\rangle .$$

If $\delta = 0$, by Theorem 6 we have that $\mathcal{D}$ will obtains a pair $(\hat{r}, \mathcal{O}(\hat{r}))$ satisfying $\hat{r} \in R_{s,\hat{Y}}$, which means that $\mathcal{D}$ will outputs 1 with probability negligibly close to 1. If $\delta = 1$, by Theorem 6 we have that in step 3.2(b) $\mathcal{P}$ can compute a state within negligible trace distance from the following state

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r = r_1 \ldots r_n \in R_{s,\hat{Y}}, \\ \hat{X}_{r,\hat{Y}} = \hat{x}_{r_1,\hat{y}_1} \ldots \hat{x}_{r_n,\hat{y}_n} \in \mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle ,$$

which is within negligible trace distance from the following state obtained by $\mathcal{P}$ after executing the generation phase of Protocol 1. As $\mathcal{D}$ and $\mathcal{P}$ will execute the verification phase of Protocol 1 with the above state as input, the probability that *accept* $= 1$ in step 3.2(c) is negligibly close to 1 by Theorem 4. This completes the proof. □

Protocol 3

Inputs: The classical algorithm $\mathcal{V}$ has classical access to $\mathcal{O}$. The quantum algorithm $\mathcal{P}$ has quantum access to $\mathcal{O}$.

Description: This is the full description of protocol.

1. $\mathcal{D}$ executes the generation phase of Protocol 1 with $\mathcal{P}$. After this, $\mathcal{D}$ will holds a string $s \in \{0,1\}^n$, a set $KT = \{(k_1, t_i)\}$ of keys and trapdoors and a set $\hat{Y} = \{\hat{y}_1, \ldots, \hat{y}_n\}$. Let $\hat{x}_{r_i,y_i} = \mathsf{INV}_{\mathcal{F}}(t_i, r_i, \hat{y}_i)$ for any $r_i \in \{0,1\}$ if $s_i = 1$, otherwise $(\hat{r}_i, \hat{x}_{r_i,\hat{y}_i}) = \mathsf{INV}_{\mathcal{G}}(t_i, \hat{y}_i)$. Then, $\mathcal{P}$ will hold a state within negligible trance distance of the following state:

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r = r_1 \ldots r_n \in R_{s,\hat{Y}}, \\ \hat{X}_{r,\hat{Y}} = \hat{x}_{r_1,\hat{y}_1} \ldots \hat{x}_{r_n,\hat{y}_n} \in \mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle.$$

   where $R_{s,\hat{Y}} = \{r = r_1 \ldots r_n \in \{0,1\}^n : s_i = 0 \wedge r_i = \hat{r}_i\}$.
2. $\mathcal{P}$ makes quantum RO query with the register containing r as inputs, which will result in a state within negligible trance distance of the following state:

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r = r_1 \ldots r_n \in R_{s,\hat{Y}}, \\ \hat{X}_{r,\hat{Y}} = \hat{x}_{r_1,\hat{y}_1} \ldots \hat{x}_{r_n,\hat{y}_n} \in \mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle |\mathcal{O}(r)\rangle.$$

3. $\mathcal{D}$ randomly chooses a bit $\delta \in \{0,1\}$, and executes Protocol 2 with $\mathcal{P}$ to measure the $(n+m)$-qubits of the above state containing r and $\mathcal{O}(r)$.
   3.1 In the case $\delta = 0$, $\mathcal{D}$ will obtain a pair $(\hat{r}, \hat{h}) \in \{0,1\}^n \times \{0,1\}^m$. Then, it outputs 0 and aborts if $\hat{r} \notin R_{s,\hat{Y}}$ or $\hat{h} \neq \mathcal{O}(\hat{r})$. Otherwise, it outputs 1.
   3.2 In the case $\delta = 1$, $\mathcal{D}$ will obtain a set $KT' = \{(k'_i, t'_i)\}_{i \in \{1,\ldots,n+m\}}$ of keys and trapdoors; $\mathcal{P}$ will obtain a state $|\phi'\rangle$ Then,
      (a) $\mathcal{V}$ sends $KT' = \{(k'_i, t'_i)\}_{i \in \{1,\ldots,n+m\}}$ to $\mathcal{P}$;
      (b) $\mathcal{P}$ computes a state within negligible trace distance of the state

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r = r_1 \ldots r_n \in R_{s,\hat{Y}}, \\ \hat{X}_{r,\hat{Y}} = \hat{x}_{r_1,\hat{y}_1} \ldots \hat{x}_{r_n,\hat{y}_n} \in \mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle |\mathcal{O}(r)\rangle$$

      by using KT' and $|\phi'\rangle$ as inputs. Then, it makes quantum RO query with the registers containing r and $\mathcal{O}(r)$ as inputs. This will lead to a state within negligible trace distance of the following state

$$\frac{1}{\sqrt{2^{\mathsf{hw}(s)}}} \sum_{\substack{r = r_1 \ldots r_n \in R_{s,\hat{Y}}, \\ \hat{X}_{r,\hat{Y}} = \hat{x}_{r_1,\hat{y}_1} \ldots \hat{x}_{r_n,\hat{y}_n} \in \mathcal{X}^n}} |r\rangle \left|\hat{X}_{r,\hat{Y}}\right\rangle$$

      (c) $\mathcal{D}$ executes the verification phase of Protocol 1 with $\mathcal{P}$ using the above state as input. Set *accept* be the output of $\mathcal{D}$ in this execution.

Outputs: $\mathcal{D}$ output *accept*.

Fig. 5. A Proof of Quantum Access to a RO

Theorem 9. *Let $\mathcal{O}^q = (\mathcal{O}^q_{priv}(\cdot), \mathcal{O}^q_{pub}(\cdot)) \xleftarrow{\$} \mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ be a QRO for some integers $n = \mathrm{poly}(\lambda)$ and $m \geq 2$. If $\mathcal{F}$ is an extended NTCF family, and $\mathcal{G}$ is the corresponding trapdoor injective family. Then, for any polynomial time quantum algorithm $\mathcal{P}^{\mathcal{O}^q_{priv}(\cdot)}$, and sufficiently large n, the probability that $\mathcal{D}^{\mathcal{O}^q_{priv}(\cdot)}$ outputs 1 is at most $3/4 + \mathrm{negl}(\kappa)$.*

Proof. Let E be an event that $\mathcal{P}$ makes a classical query with some $\hat{r} \in R_{s,\hat{Y}}$ at the end of executing Protocol 2 in step 3. By Theorem 7, we have that $|\Pr[E|\delta = 0] - \Pr[E|\delta = 1]| \leq \mathrm{negl}(\kappa)$ because $\mathcal{P}$ cannot obtain any useful information about δ holding by $\mathcal{D}$.

Let acc_δ be the event that $\mathcal{D}$ outputs 1 for a fixed $\delta \in \{0,1\}$. As δ is uniformly chosen by $\mathcal{D}$, the probability that $\mathcal{D}$ outputs 1 is equal to $\frac{\Pr[\mathrm{acc}_0]+\Pr[\mathrm{acc}_1]}{2}$. As $\mathcal{O}^q$ is a QRO, we have that $\Pr[\mathrm{acc}_0] \leq \Pr[E|\delta = 0] + \frac{1}{2}$. Moreover, by Theorem 5 we have that $\Pr[E|\delta = 1, \mathrm{acc}_1] = \mathrm{negl}(\lambda)$, as otherwise there is a polynomial algorithm which runs Protocol 2 internally before executing the verification phase of Protocol 1. By the law of total probability, we have that $\Pr[E|\delta = 1] = \Pr[E|\delta = 1, \mathrm{acc}_1]\Pr[\mathrm{acc}_1] + \Pr[E|\delta = 1, \neg\,\mathrm{acc}_1]\Pr[\neg\,\mathrm{acc}_1]$. As $\Pr[\mathrm{acc}_1] \leq 1$, we have that $\Pr[E|\delta = 1] = \Pr[E|\delta = 1, \neg\,\mathrm{acc}_1]\Pr[\neg\,\mathrm{acc}_1] + \mathrm{negl}(\lambda)$. This means that $\Pr[E|\delta = 0] = \Pr[E|\delta = 1, \neg\,\mathrm{acc}_1]\Pr[\neg\,\mathrm{acc}_1] + \mathrm{negl}(\lambda)$. Thus, $\Pr[\mathrm{acc}_0] + \Pr[\mathrm{acc}_1] =$

$$1/2 + \Pr[E|\delta = 1, \neg\,\mathrm{acc}_1]\Pr[\neg\,\mathrm{acc}_1] + \Pr[\mathrm{acc}_1] + \mathrm{negl}(\lambda) \leq 3/2 + \mathrm{negl}(\lambda).$$

In other words, the probability that $\mathcal{D}$ outputs 1 is at most $\frac{\Pr[\mathrm{acc}_0]+\Pr[\mathrm{acc}_1]}{2} \leq \frac{3}{4} + \mathrm{negl}(\lambda)$. □

5 Committed-Programming Reductions

Let $\mathcal{O}(\cdot) \xleftarrow{\$} \mathcal{F}_{n,m}$ be a RO. A cryptographic problem equipped with the RO $\mathcal{O}(\cdot)$ consisting of three algorithms $\mathrm{P} = (\mathsf{IGen}^{\mathcal{O}(\cdot)}, \mathsf{Orcl}^{\mathcal{O}(\cdot)}, \mathsf{Vrfy}^{\mathcal{O}(\cdot)})$. Specifically, the instance generator $\mathsf{IGen}^{\mathcal{O}(\cdot)}$ is a polynomial time algorithm which takes as input the security parameter κ, outputs an instance x and a secret value s_x, i.e., $(x, s_x) \leftarrow \mathsf{IGen}^{\mathcal{O}(\cdot)}(1^\kappa)$. The stateful oracle algorithm $\mathsf{Orcl}^{\mathcal{O}(\cdot)}(s_x, \cdot)$ takes as input a query $q \in \{0,1\}^*$, returns a response r to q. The deterministic verification algorithm $\mathsf{Vrfy}^{\mathcal{O}(\cdot)}$ takes as inputs an instance x, a secret value s_x and a candidate solution w, returns 1 if w is a correct solution of x, and 0 otherwise . Finally, we note that the oracle $\mathsf{Orcl}^{\mathcal{O}(\cdot)}(s_x, \cdot)$ may accept many different types of queries (i.e., it is not necessarily restricted to a single functionality), and any polynomial number of ROs can also be easily simulated by using a single one, e.g., "a query (i, h) to a RO" can be treated as "a query h to the i-th RO".

Definition 6 (Hard Problem). *Let $\mathcal{O}(\cdot) \xleftarrow{\$} \mathcal{F}_{n,m}$ be a RO. A cryptographic problem $\mathrm{P} = (\mathsf{IGen}^{\mathcal{O}(\cdot)}, \mathsf{Orcl}^{\mathcal{O}(\cdot)}, \mathsf{Vrfy}^{\mathcal{O}(\cdot)})$ is said to be hard with respect to a threshold function $t(\cdot)$, if for all polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in the game with a challenger $\mathcal{C}$ who provides inputs to $\mathcal{A}$ and answers $\mathcal{A}$'s queries to the $\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ and the RO $\mathcal{O}(\cdot)$ is negligible in the security parameter κ:*

$$\mathrm{Adv}_{\mathrm{P},\mathcal{A}}(1^\kappa) = \Pr\left[\begin{array}{c} \mathcal{O}(\cdot) \xleftarrow{\$} \mathcal{F}_{n,m} \\ (x, s_x) \leftarrow \mathsf{IGen}^{\mathcal{O}(\cdot)}(1^\kappa) \\ w \leftarrow \mathcal{A}^{\mathsf{Orcl}^{\mathcal{O}(\cdot)}(s_x,\cdot),\mathcal{O}(\cdot)}(1^\kappa, x) \end{array} : \mathsf{Vrfy}^{\mathcal{O}(\cdot)}(x, s_x, w) = 1\right] - t(\kappa).$$

Typically, we have $t(\kappa) = 0$ for computational problems, and $t(\kappa) = 1/2$ for decisional problems. In this paper, we only consider *falsifiable* problems, which require that $\mathsf{IGen}, \mathsf{Orcl}$ and Vrfy are polynomial time algorithms. Since a standard cryptographic problem can be seen as a problem $\mathrm{P} = (\mathsf{IGen}^{\mathcal{O}(\cdot)}, \mathsf{Orcl}^{\mathcal{O}(\cdot)}, \mathsf{Vrfy}^{\mathcal{O}(\cdot)})$ where the three algorithms do not use the RO $\mathcal{O}(\cdot)$, Definition 6 captures many cryptographic problems such as SIS and CCA/CPA-secure PKE (in the ROM).

A black-box (BB) reduction $\mathcal{R}$ from a problem P_1 to another problem P_2 (in the ROM) is a polynomial time oracle algorithm such that $\mathcal{R}^{\mathcal{A}}$ solves P_1 whenever $\mathcal{A}$ solves P_2. A BB-reduction $\mathcal{R}$ in the ROM can exploit various properties (e.g., observability and programmability) of the RO to answer the queries to the "crypto-oracle" $\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ (e.g., the signing oracle) from $\mathcal{A}$.

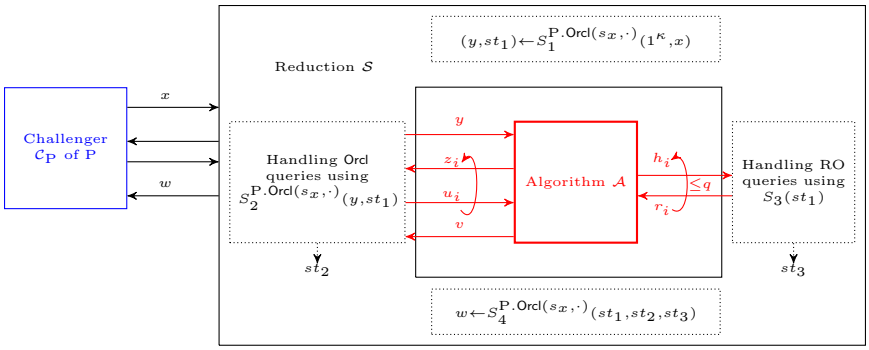

Fig. 6. A (F, I)-splittable reduction $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ from P to Q, where $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_4$ can have access to $\mathrm{P.Orcl}(s_x, \cdot)$, but $\mathcal{S}_3$ does not access $\mathrm{P.Orcl}(s_x, \cdot)$.

5.1 Definitions

In the following, we focus on the class of black-box reductions in the ROM that only apply the following two strategies to simulate ROs for answering the crypto-oracle queries: 1) the reduction simulates the RO in a particular way such that the adversary may distinguish the simulated RO from a real one, but the crypto-oracle queries can be successfully answered w.r.t. the simulated RO if certain conditions on the adversary's queries are satisfied; 2) the reduction perfectly simulates the RO for the adversary, but it will deviate from the simulated ROs in answering some crypto-oracle queries by (implicitly) replacing the responses of the RO at some unknown points (to which it cannot directly compute the responses) with random values, which means that the adversary can detect this inconsistency by making a RO query with any one of those points. As a problem may involve many ROs, we allow the reduction to apply different simulation strategies for different ROs (but only apply one of the above two strategies to each RO). We distinguish the ROs for a given reduction $\mathcal{S}$ into two types, and say that a RO is a Type-I RO if $\mathcal{S}$ will apply the first RO simulation strategy to it, otherwise a Type-II RO if $\mathcal{S}$ will apply the one to it. Note that for a fixed $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ and query z, when and how $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ uses the RO $\mathcal{O}(\cdot)$ to compute a response to z is also fixed, one can count the RO queries required by $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ in producing the response, and use an index function (taking z as input) to specify the positions of the "bad" Type-II RO queries that deviate from the simulation of the Type-II ROs in this computation.

In Definition 7, we define a splittable property for a BB-reduction in the ROM, which will be used to formalize the notion of committed-programming reduction in Definition 8. In both definitions, we will omit the superscript $\mathcal{O}(\cdot)$ and denote problem $\mathrm{P} = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ for simplicity of exposition. Intuitively, the splittable property says that the simulation of $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ and the RO $\mathcal{O}(\cdot)$ can be done by using two sub-algorithms that do not interact with each other. We associate a splittable reduction with a pair of deterministic functions (F, I), where $F(\cdot, \cdot)$ is a function for answering Type-I RO queries, and $I(\cdot)$ is an index function specifying the position of the "bad" Type-II RO queries in computing the response to a crypto-oracle query. As a reduction may only have Type-I (resp., Type-II) RO, we allow dummy $I(\cdot) = \emptyset$ (resp., $F = \perp$).

Definition 7 (Splittability). *Using the notations above, a BB-reduction $\mathcal{S}$ from problem $\mathrm{P} = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ to $\mathrm{Q} = (\mathsf{IGen}^{\mathcal{O}(\cdot)}, \mathsf{Orcl}^{\mathcal{O}(\cdot)}, \mathsf{Vrfy}^{\mathcal{O}(\cdot)})$ in the ROM is (F, I)-splittable if for any $\mathcal{A}$ solving Q with at most q RO queries, $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ can be split into four sub-algorithms (as depicted in Fig. 6):*

1. *Given a security parameter κ and an instance x of P as inputs, $\mathcal{S}$ first runs sub-algorithm $\mathcal{S}_1^{\mathrm{P.Orcl}(s_x,\cdot)}(1^\kappa, x)$ to generate an instance y of Q and a state $st_1 = (\tau, f)$ consisting of a string τ for Type-I RO and a real random function $f(\cdot)$ for Type-II RO, i.e., $(y, st_1) \leftarrow \mathcal{S}_1^{\mathrm{P.Orcl}(s_x,\cdot)}(1^\kappa, x)$, which are then used as inputs to run $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ and $\mathcal{S}_3(st_1)$ to interact with $\mathcal{A}$;*

2. *The sub-algorithm* $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ *takes an instance* y *of* Q *and a state* $st_1 = (\tau, f)$ *as inputs, invokes a single instance of* $\mathcal{A}$ *with input* y*:*
   - *Whenever receiving a* $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ *query* z_i *from* $\mathcal{A}$, $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ *may abort if certain condition on* z_i *is not satisfied for the simulation of Type-I RO. Otherwise, let* $\{h_{i,j}\}_{j\in\{1,\ldots,q_i\}}$ *be all possible RO queries (indexed by the order of the query) required for computing a response* u_i *to* z_i*. Then, a set* $\{r_{i,j}\}_{j\in\{1,\ldots,q_i\}}$ *is used as the RO responses to* $\{h_{i,j}\}_{j\in\{1,\ldots,q_i\}}$ *for generating* u_i *such that*
     - *If* $h_{i,j}$ *is a Type-I RO query,* $r_{i,j} = F(\tau, h_{i,j})$*;*
     - *Else if* $h_{i,j}$ *is a Type-II RO query and* $j \in I(z_i)$*, randomly choose* $r_{i,j}$ *with consistency (i.e., the same* $r_{i,j}$ *is chosen for the same* $h_{i,j}$ *during the whole simulation),*[7] *otherwise* $r_j = f(h_{i,j})$*;*
   - *Whenever* $\mathcal{A}$ *outputs a solution* v *of* y *at some time,* $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}$ *may abort if certain condition on* v *is not satisfied for the simulation of Type-I RO, otherwise outputs a state* st_2 *and terminates;*
3. *The sub-algorithm* $\mathcal{S}_3(st_1)$ *takes a state* $st_1 = (\tau, f)$ *as input, randomly chooses an integer* $k^* \xleftarrow{\$} \{1, \ldots, q_2\}$*, where* q_2 *is the maximum number of Type-II RO queries from* $\mathcal{A}$*. Whenever receiving a RO query* h *from* $\mathcal{A}$*,* $\mathcal{S}_3$ *outputs* $st_3 = h$ *and terminates if* h *is the* k^**-th Type-II RO query. Otherwise, answer* $\mathcal{A}$ *with* $r = F(\tau, h)$ *if* h *is a Type-I RO query, or* $r = f(h)$ *if* h *is a Type-II RO query. If* $\mathcal{A}$ *terminates,* $\mathcal{S}_3$ *outputs* $st_3 = \perp$ *and terminates;*
4. *Finally,* $\mathcal{S}$ *computes* $w \leftarrow \mathcal{S}_4^{\mathrm{P.Orcl}(s_x,\cdot)}(st_1, st_2, st_3)$ *as his own solution of* x*.*

Note that $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ and $\mathcal{S}_3(st_1)$ are the exact parts of $\mathcal{S}$ directly interacting with the algorithm $\mathcal{A}$ (as depicted in Fig. 6), and solely try to answer the queries from (a single instance of) $\mathcal{A}$. In particular, a splittable reduction $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ will not rewind $\mathcal{A}$. Moreover, $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ and $\mathcal{S}_3(st_1)$ only share the common input st_1, and will not interact with each other after being invoked. We finally note that $\mathcal{S}_3$ does not have access to $\mathrm{P.Orcl}(s_x, \cdot)$, which is necessary for isolating the behaviors of $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}$ and $\mathcal{S}_3$.

For a successful reduction $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$, we need to consider: 1) the difference between the simulated Type-I RO $\mathcal{O}_{F_\lambda}(\cdot) = F(st_1, \cdot)$ and a real RO $\mathcal{O}(\cdot)$ in the adversary's view. For this, we will use a notation $\gamma(q, \mathcal{O}_{F_\lambda})$:

$$\gamma(q, \mathcal{O}_{F_\lambda}) = \max_{\mathcal{A}} \left| \Pr\left[\mathcal{A}^{\mathcal{O}_{F_\lambda}(\cdot)}(1^\kappa) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}(\cdot)}(1^\kappa) = 1\right] \right|,$$

where q is a positive integer, and the maximum is taken over all algorithms $\mathcal{A}$ making q queries; 2) the event E_1 that $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ aborts because certain condition is not satisfied for the simulation of Type-I RO. A non-trivial reduction has to make sure that the probability $\Pr[\neg E_1] = 1 - \Pr[E_1]$ is noticeable; 3) the event E_2 that $\mathcal{A}$ makes a "bad" Type-II RO query $h_{i,j}$ to $\mathcal{S}_3(st_1)$ such that $h_{i,j}$ is the j-th RO query required for computing a response to some $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z_i and $j \in I(z_i) \neq \emptyset$ (and thus may detect the inconsistence between the behavior of $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ and the simulated ROs using $\mathcal{S}_3$). Usually, a non-trivial adversary is ensured to make such a "bad" Type-II query with $h_{i,j}$ such that the reduction can obtain the unknown $h_{i,j}$ for solving its own problem. By definition, we have $\Pr[E_1] = 0$ (or $\Pr[E_2] = 0$, respectively) if there is no Type-I (or Type-II, respectively) RO.

Definition 8 (Committed-Programming Reduction). *Using the notations of Definition 7 and above, we say that problem* $\mathrm{P} = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ *can be reduced to* $\mathrm{Q} = (\mathsf{IGen}^{\mathcal{O}(\cdot)}, \mathsf{Orcl}^{\mathcal{O}(\cdot)}, \mathsf{Vrfy}^{\mathcal{O}(\cdot)})$ *under* $(\{F_\lambda\}_{\lambda\in(0,1)}, I)$*-Committed-Programming Reduction, or* $(\{F_\lambda\}_{\lambda\in(0,1)}, I)$*-CPRed in brief, if for any* $\lambda \in (0, 1)$*, and for any (even unbounded) algorithm* $\mathcal{A}$ *solving* Q *with non-negligible advantage* ϑ_A *by making* q_1 $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ *queries and* q_2 *RO queries, there is a* (F_λ, I)*-splittable reduction* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ *such that*

1. *there exists a fixed polynomial* $\mathsf{poly}_1(\cdot)$ *such that* $\Pr[\neg E_1] > \lambda - \mathsf{poly}_1(q_1)\lambda^2$*;*
2. *there exists a fixed polynomial* $\mathsf{poly}_2(\cdot)$ *such that* $\gamma(q_3, \mathcal{O}_{F_\lambda}) \leq \mathsf{poly}_2(q_3)\lambda^2$ *for any positive integer* q_3 *and* $\mathcal{O}_{F_\lambda}(\cdot) = F_\lambda(\tau, \cdot)$*;*

[7] In the case that $j \in I(z_i)$, the algorithm $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}$ may not really pick r_j as both values $(h_{i,j}, r_j)$ might come from its oracle $\mathrm{P.Orcl}(s_x, \cdot)$, and are unknown to him.

3. *given a* $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ *query* z*, except at the Type-II RO queries with positions specified by* $I(z)$*, the behavior of* $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ *in computing a response to* z*, conditioned on* E_1 *not happening, is statistically close to that of the real* $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ *with the RO* $\mathcal{O}(\cdot)$ *being changed to the simulated one by* $\mathcal{S}_3$*;*
4. *if* $I(\cdot) = \emptyset$*, the advantage that* $\mathcal{A}$ *outputs a solution* v *of* y *to* $\mathcal{S}_2^{\mathrm{POrcl}(s_x,\cdot)}(y, st_1)$*, conditioned on* E_1 *not happening, is at least* $\vartheta_A - \gamma(q_3, \mathcal{O}_{F_\lambda})$ *for some integer* q_3*. Moreover, given a solution* v *of* y*,* $\mathcal{S}_4^{\mathrm{P.Orcl}(s_x,\cdot)}$ *can find a solution* w *of* x*;*
5. *if* $I(\cdot) \neq \emptyset$*, the advantage that* $\mathcal{A}$ *outputs a solution* v *of* y *to* $\mathcal{S}_2^{\mathrm{POrcl}(s_x,\cdot)}(y, st_1)$*, conditioned on* E_1 *not happening, is negligible. Moreover, given a "bad" Type-II RO query* $h_{i,j}$ *for some* $j \in I(z_i)$*,* $\mathcal{S}_4^{\mathrm{P.Orcl}(s_x,\cdot)}$ *can find a solution* w *of* x*.*

Informally, the first two conditions say that we can set the parameter λ such that 1) $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ does not abort with noticeable probability; 2) the simulated Type-I RO $\mathcal{O}_{F_\lambda}(\cdot)$ is not far from a real RO (note that the simulation of the Type-II RO using a random function $f(\cdot)$ is identical to a real RO). The third condition implies that if the RO queries from $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ are all answered with the simulated RO by $\mathcal{S}_3$, then the behavior of $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$, conditioned on $E_1 \vee E_2$ not happening, are statistically close to that of the real $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ in the adversary's view. The last two are the successful condition of $\mathcal{S}$. In particular, the last one says that no adversary can win the game simulated by $\mathcal{S}$, which implies that any non-trivial adversary will always distinguish the simulated game from a real one by making a "bad" Type-II RO query (i.e., E_2 will always happen for any non-trivial adversary), and any "bad" Type-II RO query is sufficient for $\mathcal{S}$ to solve its own problem P.

To better elaborate the notions of CPReds, we will give some concrete examples of CPReds for some well-known existing schemes in next subsection.

5.2 Examples of CPReds

We now take the full-domain hash (FDH) signature [4] and the Boneh-Franklin IBE scheme [8] as concrete examples to explain the notion of CPReds.

The FDH Signature. Let $\mathsf{Perm}(\cdot,\cdot) : \mathcal{K} \times \{0,1\}^\ell \to \{0,1\}^\ell$ be some family of trapdoor permutations (TDP): given a permutation index $s \in \mathcal{K}$, $\mathsf{Perm}(s,\cdot)$ is a permutation. Let $H : \{0,1\}^* \to \{0,1\}^\ell$ be a full-domain hash function modeled as a RO. We first briefly recall the FDH signature from TDP, which consists of three algorithms $\Pi_{\mathrm{FDH}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$.

- $\mathsf{KeyGen}(1^\kappa)$: the key generation algorithm takes the security parameter κ as input, randomly chooses a permutation index s together with a trapdoor td, sets $vk = s$ as the verification key and $sk = td$ as the signing key, i.e., $(vk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$.
- $\mathsf{Sign}(td, \mu)$: given the signing key sk and a message $\mu \in \{0,1\}^*$, the signing algorithm computes $H(\mu) \in \{0,1\}^\ell$ by using the hash function H (modeled as a RO) and $x \in \{0,1\}^\ell$ such that $\mathsf{Perm}(s,x) = H(\mu)$ by using the trapdoor td, return $\sigma = x$ as the signature on message μ, i.e., $\sigma \leftarrow \mathsf{Sign}(td, \mu)$.
- $\mathsf{Verify}(vk, \mu, \sigma)$: the verification algorithm takes as inputs $vk = s$, a message μ and a signature $\sigma = x$, returns 1 if $\mathsf{Perm}(s,x) = H(\mu)$, and 0 otherwise, i.e., $1/0 \leftarrow \mathsf{Verify}(vk, \mu, \sigma)$.

The goal of a reduction for the FDH signature is to break the one-wayness of TDP: given a permutation index s^* and a uniformly random $y^* \xleftarrow{\$} \{0,1\}^\ell$, output a preimage $x^* \in \{0,1\}^\ell$ such that $y^* = \mathsf{Perm}(s^*, x^*)$. We now show that for any adversary $\mathcal{A}$ breaking the existential unforgeability against chosen message attacks (i.e., EUF-CMA, see Section C in the supplement material) of the FDH signature with non-negligible advantage ϑ_A by making q_1 signing queries and q_2 RO queries, there is a $(\{F_\lambda\}_{\lambda\in(0,1)}, \emptyset)$-CPRed $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ breaking the one-wayness of TDP, which is implicit in [4,7,35]. Specifically, for any constant $\lambda \in (0,1)$, pick positive integers p_1, p such that $\bar{\lambda} = p_1/p$ and $\lambda - \bar{\lambda} \leq \lambda^2$. Given a TDP challenge instance (s^*, y^*) as inputs, $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ works as follows:

- $(s^*, st_1 = (f_1, f_2, s^*, y^*)) \leftarrow \mathcal{S}_1(1^\kappa, (s^*, y^*))$, where $f_1 : \{0,1\}^* \to \{1, \ldots, p\}$ and $f_2 : \{0,1\}^* \to \{0,1\}^\ell$ are two random functions (which can be instantiated with k-wise independent functions [35]. We directly use "random functions" to simplify the analysis);
- $\mathcal{S}_2(s^*, st_1)$ uses $vk = s^*$ as the challenge verification key to invoke $\mathcal{A}$. For a signing query $z_i \in \{0,1\}^\ell$ from $\mathcal{A}$, $\mathcal{S}_2$ outputs $st_2 = \perp$ and aborts if $f_1(z_i) \leq p_1$, and otherwise returns $u_i = f_2(z_i) \in \{0,1\}^\ell$ to $\mathcal{A}$. Whenever $\mathcal{A}$ outputs a forgery $v = (z^*, u^*)$ and terminates, $\mathcal{S}_2$ outputs $st_2 = \perp$ and aborts if $f_1(z^*) > p_1$, otherwise outputs $st_2 = (z^*, u^*)$ and terminates;
- $\mathcal{S}_3(st_1)$ uses $F_\lambda(st_1, \cdot)$ to answer a RO query z from $\mathcal{A}$:

$$F_\lambda(st_1, z) = \begin{cases} y^*, & \text{if } f_1(z) \leq p_1; \\ \mathsf{Perm}(s^*, f_2(z)), & \text{otherwise.} \end{cases}$$

  Whenever $\mathcal{A}$ terminates, $\mathcal{S}_3$ outputs $st_3 = \perp$ and terminates;
- $\mathcal{S}_4(st_1, st_2, st_3)$ outputs $w^* = u^*$ if $st_2 = (z^*, u^*) \wedge \mathsf{Perm}(s^*, u^*) = y^*$, otherwise outputs a random $x \xleftarrow{\$} \{0,1\}^\ell$.

We first note that for the above reduction $\mathcal{S}$, there is only a Type-I RO H. Let E_1 be the event that $\mathcal{S}_2$ aborts at some time. Then we have the following facts:

1. As f_1 is a random function, we have $\Pr[\neg E_1] \geq \bar{\lambda}(1-\bar{\lambda})^{q_1}$. Using the fact that $(1-\bar{\lambda})^{q_1} \geq 1 - q_1\bar{\lambda}$ and $\lambda < 1$, we have $\Pr[\neg E_1] \geq \bar{\lambda}(1 - q_1\bar{\lambda}) \geq \lambda - (2q_1+1)\lambda^2$;
2. As f_2 is a random function and $y^* \in \{0,1\}^\ell$ is uniformly random, any algorithm making q_3 queries can only distinguish the simulated RO $\mathcal{O}_{F_\lambda}(\cdot) = F_\lambda(st_1, \cdot)$ from a real RO $\mathcal{O}(\cdot)$ by making two queries z_1, z_2, s.t., $F_\lambda(st_1, z_1) = F(st_1, z_2) = y^*$ (i.e., $f_1(z_1) \leq p_1 \wedge f_1(z_2) \leq p_1$). As f_1 is a random function, for any q_3 RO queries, the probability that there is at most one query hitting y^* is $(1-\bar{\lambda})^{q_3} + q_3\bar{\lambda}(1-\bar{\lambda})^{q_3-1} = (1 + (q_3-1)\bar{\lambda})(1-\bar{\lambda})^{q_3-1}$. Using the fact that $(1-\bar{\lambda})^{q_3-1} \geq 1 - (q_3-1)\bar{\lambda}$ and $\lambda - \bar{\lambda} \leq \lambda^2$, we have

$$\gamma(q_3, \mathcal{O}_{F_\lambda}) \leq 1 - (1-\bar{\lambda})^{q_3} - q_3\bar{\lambda}(1-\bar{\lambda})^{q_3-1} \leq (q_3-1)^2\bar{\lambda}^2 \leq 2(q_3-1)^2\lambda^2;$$

3. the behavior of $\mathcal{S}_2^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}(y, st_1)$ in answering a signing query, conditioned on E_1 not happening, is identical to that of the real signing oracle with the RO $H(\cdot)$ being changed to the simulated one by $\mathcal{S}_3(st_1)$.
4. the advantage that $\mathcal{A}$ outputs a valid forgery (z^*, u^*) (i.e., $\mathsf{Perm}(s^*, u^*) = y^*$), conditioned on E_1 not happening, is at least $\vartheta_A - 2(q_3-1)^2\lambda^2$ for $q_3 = q_1 + q_2$, as the adversary will trigger at most q_3 RO queries. Moreover, given a forgery (z^*, u^*), $\mathcal{S}_4^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}$ will output u^*, s.t., $\mathsf{Perm}(s^*, u^*) = y^*$.

Clearly, the advantage of $\mathcal{S}$ in outputting u^* satisfying $\mathsf{Perm}(s^*, u^*) = y^*$ is at least $(\vartheta_A - 2(q_3-1)^2\lambda^2) \cdot \Pr[\neg E_1] \geq \lambda\vartheta_A - (2(q_3-1)^2 + 2q_1 + 1)\lambda^2$. By setting $\mathrm{poly}_3(q_1, q_2) = 2(q_1+q_2-1)^2 + 2q_1 + 1$ and $\lambda = \vartheta_A/(2\mathrm{poly}_3(q_1, q_2))$, the advantage of $\mathcal{S}$ is at least $\vartheta_A^2/(4\mathrm{poly}_3(q_1, q_2))$, which is non-negligible if ϑ_A is non-negligible.[8] This shows that $\mathcal{S}$ is a $(\{F_\lambda\}_{\lambda\in(0,1)}, \emptyset)$-CPRed for the EUF-CMA security of the FDH signature from TDP [4]. Besides, the variant PSF-FDH signature where the TDP is replaced with preimage sampleable trapdoor functions (PSF), and the GPV-IBE from LWE [18] are also provably secure under $(\{F_\lambda\}_{\lambda\in(0,1)}, \emptyset)$-CPReds, which can be confirmed by inspection of the reductions given in [18] and [7,35].

The Boneh-Franklin IBE. We now take the IND-CPA security of the Boneh-Franklin IBE (BF-IBE) scheme [8] as an example to elaborate a bit more on the notion of CPRed (especially on the role of $I(\cdot)$). We first briefly recall the BF-IBE scheme, which consists of four algorithms $\Pi_{\text{BF-IBE}} = (\mathsf{KeyGen}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$. Let $\mathbb{G}_1, \mathbb{G}_2$ be two groups of prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map and let g be a generator of $\mathbb{G}_1$. Let $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ be two hash functions (both modeled as ROs).

[8] Note that one can use a more precise argument to obtain a tighter argument. Here, we only focus on showing that $\mathcal{S}$ is a CPRed.

- $\mathsf{KeyGen}(1^\kappa)$: The key generation algorithm takes the security parameter κ as input, randomly chooses $s \xleftarrow{\$} \mathbb{Z}_p^*$, outputs the master public key $mpk = (g, h = g^s)$ and master secret key $msk = s$, i.e., $(mpk, msk) \leftarrow \mathsf{KeyGen}(1^\kappa)$.
- $\mathsf{Extract}(msk, id)$: The user key extraction algorithm takes $msk = s$ and an identity $id \in \{0,1\}^*$ as inputs, computes $u_{id} = H_1(id) \in \mathbb{G}_1$ and outputs the user secret key $usk_{id} = u_{id}^s$, i.e., $usk_{id} \leftarrow \mathsf{Extract}(msk, id)$.
- $\mathsf{Enc}(mpk, id, \mu)$: The encryption algorithm takes $mpk = (g, h)$, an identity $id \in \{0,1\}^*$ and message $\mu \in \{0,1\}^\ell$ as inputs, first compute $u_{id} = H_1(id)$. Then, it randomly chooses $r \xleftarrow{\$} \mathbb{Z}_p^*$, computes $c_1 = g^r, R = e(u_{id}, h)^r = e(u_{id}, g)^{rs}$ and $c_2 = \mu \oplus H_2(R)$. Finally, outputs the ciphertext $C_{id} = (c_1, c_2)$, i.e., $C_{id} \leftarrow \mathsf{Enc}(mpk, id, \mu)$.
- $\mathsf{Dec}(usk_{id}, C_{id})$: The decryption algorithm takes a user secret key $usk_{id} = u_{id}^s$ for some identity $id \in \{0,1\}^*$ and a ciphertext $C_{id} = (c_1 = g^r, c_2)$ encrypted using identity id as inputs, computes and returns $\mu = c_2 \oplus H_2(R)$ where $R = e(u_{id}^s, g^r) = e(u_{id}, g)^{rs}$, i.e., $\mu \leftarrow \mathsf{Dec}(usk_{id}, C_{id})$.

The goal of a reduction for the semantic security (i.e., IND-ID-CPA security, see Section D in the supplement material) of the BF-IBE scheme is to solve the BDH problem: given (g, g^a, g^b, g^c) with uniformly random $a, b, c \xleftarrow{\$} \mathbb{Z}_q^*$ as inputs, output $e(g,g)^{abc}$. In order to use the ability of the adversary attacking the IND-ID-CPA security of the BF-IBE scheme to solve the BDH problem, the reduction has to answer two types of crypto-oracles queries (i.e., user key extraction query and challenge ciphertext query), and two ROs (i.e., H_1 and H_2) queries from the adversary. We now show that for any adversary $\mathcal{A}$ breaking the semantic security of the BF-IBE scheme with advantage ϑ_A by making q_1 user key extraction queries, a single challenge ciphertext query, $q_{2,1}$ queries to H_1 and $q_{2,2}$ queries to H_2 (i.e., the total RO queries is $q_2 = q_{2,1} + q_{2,2}$), there is a (F, I)-CPRed $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ solving the BDH problem, which is implicit in [8]. As before, for any constant $\lambda \in (0,1)$, pick positive integers p_1, p such that $\bar{\lambda} = p_1/p$ and $\lambda - \bar{\lambda} \leq \lambda^2$. Given a BDH challenge instance $\mathsf{inst} = (g, g^a, g^b, g^c)$ as inputs, $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ works as follows:

- $(mpk = (g, g^a), st_1 = (\tau, f_2)) \leftarrow \mathcal{S}_1(1^\kappa, \mathsf{inst})$, where $\tau = (f_{1,1}, f_{1,2}, \mathsf{inst})$, $f_{1,1} : \{0,1\}^* \to \{1, \ldots, p\}$, $f_{1,2} : \{0,1\}^* \to \mathbb{Z}_p^*$ and $f_2 : \mathbb{G}_2 \to \{0,1\}^\ell$ are three random functions (this implicitly sets the master secret key $msk = a$);
- $\mathcal{S}_2((g, g^a), st_1)$ uses $mpk = (g, h = g^a)$ as the master public key to invoke $\mathcal{A}$, and answers $\mathcal{A}$'s queries as follows:
  - Whenever receiving a user key extraction query $id \in \{0,1\}^*$ from $\mathcal{A}$, if $f_{1,1}(id) \leq p_1$, output $\perp$ and abort, else compute $u_{id} = H_1(id) = g^{f_{1,2}(id)}$ (by simulating a H_1 query id), and return the user secret key $usk_{id} = u_{id}^a = h^{f_{1,2}(id)}$ to $\mathcal{A}$;
  - When receiving a challenge ciphertext query (id^*, μ_0, μ_1) for $id^* \in \{0,1\}^*$ and $\mu_0, \mu_1 \in \{0,1\}^n$, if $f_{1,1}(id^*) > p_1$, output $\perp$ and abort. Otherwise, compute $u_{id^*} = H_1(id^*) = g^b$ (by simulating a H_1 query id^*), and return the challenge ciphertext $C_{id^*} = (g^c, \mu_\delta \oplus \rho)$ by randomly choosing $\rho \xleftarrow{\$} \{0,1\}^n$ and $\delta \xleftarrow{\$} \{0,1\}$ (i.e., implicitly setting $H_2(R) = \rho$ for $R = e(u_{id^*}, h)^c = e(g,g)^{abc}$).

  Whenever $\mathcal{A}$ outputs a guess $\delta' \in \{0,1\}$ for δ and terminates, $\mathcal{S}_2$ outputs $st_2 = \perp$ and terminates;
- $\mathcal{S}_3(st_1)$ parses $st_1 = (\tau, f_2)$ and uniformly chooses an integer $k^* \xleftarrow{\$} \{1, \ldots, q_{2,2}\}$ at random. Then, it answers $\mathcal{A}$'s RO queries as follows:
  - Whenever receiving a H_1 query $id \in \{0,1\}^*$ from $\mathcal{A}$, compute and return $F_\lambda(\tau, id)$ using the following function:
$$F_\lambda(\tau, id) = \begin{cases} g^b, & \text{if } f_{1,1}(id) \leq p_1; \\ g^{f_{1,2}(id)}, & \text{if } f_{1,1}(id) > p_1 \ . \end{cases}$$
  - Whenever receiving a H_2 query $R \in \mathbb{G}_2$ from $\mathcal{A}$, if it is the k^*-th query to H_2 (i.e., $t = 2$), outputs $st_3 = R$ and terminates. Otherwise, return $f_2(R)$ to $\mathcal{A}$.

  Whenever $\mathcal{A}$ terminates, $\mathcal{S}_3$ outputs $st_3 = \perp$ and terminates;
- $\mathcal{S}_4(st_1, st_2, st_3)$ returns $w^* = R$ if $st_3 = R$, else return a random $R \xleftarrow{\$} \mathbb{G}_2$.

We first note that for the above reduction $\mathcal{S}$, there is a Type-I RO H_1 and a Type-II RO H_2. Let E_1 be the event that $\mathcal{S}_2$ aborts at some time. Let E_2 be the event that the adversary makes a "bad" H_2 query $R = e(g,g)^{abc} \in \mathbb{G}_2$. Then, we have the following facts:

1. $\Pr[\neg E_1] \geq \lambda - (2q_1+1)\lambda^2$ by the same analysis as for the FDH signature;
2. $\gamma(q_3, \mathcal{O}_{F_\lambda}) \leq 2(q_3-1)^2\lambda^2$ for any positive integer q_3 and $\mathcal{O}_{F_\lambda}(\cdot) = F_\lambda(st_1, \cdot)$ by the same analysis as for the FDH signature;
3. the behavior of $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$, conditioned on E_1 not happening, only deviates at the RO queries specified by $I(z)$ in computing the response to the challenger ciphertext query z from that of the real one with the ROs being changed to the simulated RO by $\mathcal{S}_3(st_1)$, where

$$I(query) = \begin{cases} \emptyset, & \text{if } query \text{ is a user key extraction query;} \\ 2, & \text{if } query \text{ is a challenge ciphertext query.} \end{cases}$$

   In other words, the reduction will only replace the second RO query in generating the challenge ciphertext with a random one (as it does not know $R = e(g,g)^{abc}$, and cannot normally compute $H_2(R)$).
4. the advantage that $\mathcal{A}$ outputs $\delta' = \delta$, conditioned on E_1 not happening, is 0 (as the random ρ perfectly hides δ from $\mathcal{A}$). Moreover, given a "bad" H_2 query $R = e(g,g)^{abc}$, $\mathcal{S}_4^{\mathrm{P.Orcl}(s_x,\cdot)}$ will output a valid solution R.

Using the above facts, we have $\Pr[E_2] \geq (\vartheta_A - 2(q_3-1)^2\lambda^2) \cdot \Pr[\neg E_1]$ by a simple analysis. Moreover, if E_2 happens, the probability that $\mathcal{S}$ can output $R = e(g,g)^{abc}$ is at least $1/q_{2,2}$. Thus, the advantage of $\mathcal{S}$ outputting $R = e(g,g)^{abc}$ is at least $(\vartheta_A \cdot \Pr[\neg E_1] - 2(q_3-1)^2\lambda^2)/q_{2,2} \geq (\lambda\vartheta_A - 2\lambda^2 - (2q_1+1)\lambda^2\vartheta_A)/q_{2,2}$, where $q_3 = q_1 + q_{2,1} + 1$. By setting $\mathrm{poly}(q_1, q_{2,1}) = 2q_1 + 1 + 2(q_1+q_{2,1})^2$ and $\lambda = \vartheta_A/(2\mathrm{poly}(q_1, q_{2,1}))$, the advantage of $\mathcal{S}$ is at least $\vartheta_A^2/(2q_{2,2}\mathrm{poly}(q_1, q_{2,1}))$, which is non-negligible if ϑ_A is non-negligible. This shows that $\mathcal{S}$ is a $(\{F_\lambda\}_{\lambda\in(0,1)}, I)$-CPRed for the BF-IBE scheme [8]. Besides, the "implicit rejection" variant of the KEM from TDP due to Shoup [30], and the "implicit rejection" [20] variants of the KEM from the Fujisaki-Okamoto (FO) transform [16] are also provably secure under $(\emptyset, I)$-CRPeds for some index function I.

6 Lifting CPReds to Reductions in the QROM

In this section, we show that if a problem P can be reduced to Q under CPReds associated with an instance-extraction algorithm in the ROM, then there is a reduction from P to Q in the QROM. Informally, the instance-extraction algorithm can extract an instance of problem P (and produce an auxiliary information for the RO simulation) from an instance of Q, which is needed to ensure that the sub-algorithm (of the CPRed) for simulating the RO can be "safely" quantized to simulate a quantum RO, and that one can smoothly replace a real quantum RO with a simulated one in the security proof. We note that the history reduction given in [7] also requires a similar (and possibly stronger) algorithm.

Theorem 10. *Using the notations of Definition 8, if problem* $\mathrm{P} = (\mathsf{IGen}, \mathsf{Orcl}, \mathsf{Vrfy})$ *can be reduced to* $\mathrm{Q} = (\mathsf{IGen}^{\mathcal{O}(\cdot)}, \mathsf{Orcl}^{\mathcal{O}(\cdot)}, \mathsf{Vrfy}^{\mathcal{O}(\cdot)})$ *under* $(\{F_\lambda\}_{\lambda\in(0,1)}, I)$*-CPRed, and for any constant* λ*, the* (F_λ, I)*-splittable reduction* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ *is associated with an instance-extraction algorithm* $\mathrm{Q.Extract}(\cdot,\cdot)$ *for* Q *such that for any* $(y, s_y) \leftarrow \mathrm{Q.IGen}^{\mathcal{O}(\cdot)}(1^\kappa)$ *and* $(x, s_x, st_1 = (\tau, f)) \leftarrow \mathrm{Q.Extract}(y, s_y)$*:*

- *there exists a fixed polynomial* $\mathsf{poly}_2'(\cdot)$ *such that given* y *and* s_y*, the output distribution of any (even unbounded) algorithm* $\mathcal{A}$ *making* q_3 *quantum queries to the oracle* $\mathcal{O}_{F_\lambda}(\cdot) : |h_1, h_2\rangle \to |h_1, F_\lambda(\tau, h_1) \oplus h_2\rangle$ *is at most* $\mathsf{poly}_2'(q_3)\lambda^2$ *far from the case when* $\mathcal{A}$ *is given access to* $\mathcal{O}_{f_1}(\cdot) : |h_1, h_2\rangle \to |h_1, f_1(h_1) \oplus h_2\rangle$ *with a uniform* $f_1 \xleftarrow{\$} \mathcal{F}_{n,m}$*;* [9]
- *The distribution of* (x, s_x, y, τ) *is identical to that of* (x', s_x', y', st_1') *generated by* $(x', s_x') \leftarrow \mathrm{P.IGen}(1^\kappa)$ *and* $(y', st_1') \leftarrow \mathcal{S}_1^{\mathrm{P.Orcl}(s_x',\cdot)}(1^\kappa, x)$*, over the randomness used by* $\mathrm{Q.IGen}^{\mathcal{O}(\cdot)}, \mathrm{Q.Extract}, \mathrm{P.IGen}$ *and* $\mathcal{S}_1^{\mathrm{P.Orcl}(s_x',\cdot)}$*,*

[9] The fact that $\mathcal{A}$ has the knowledge of (y, s_y) is very crucial for the proof of Theorem 10 because we need to consider an adversary that can make $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, \cdot)$ queries, and in particular it may trigger a RO query depending on the information of s_y by making some $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, \cdot)$ query.

then for any quantum polynomial time algorithm $\mathcal{A}$ against Q *using at most q_1 queries to* $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ *and q_2 quantum RO queries such that $dq_1 + q_2 \leq q_3$, there is a reduction $\mathcal{S}'$ from* P *to* Q *in the QROM, where d is the maximum number of RO queries that is required for computing a response to any* $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ *query.*

One can check that the schemes discussed in Section 5.2 have CPReds associated with natural instance-extraction algorithms. Taking the FDH signature [4] as an example, let $((vk, y^*), sk, st_1 = (f_1, f_2, vk, y^*)) \leftarrow \mathsf{Extract}(vk, sk)$ be an algorithm which takes a verification key vk and a signing key sk as inputs, outputs (vk, y^*), sk and $st_1 = (f_1, f_2, vk, y^*)$ by randomly choosing $y^* \xleftarrow{\$} \{0,1\}^\ell$, $f_1 : \{0,1\}^* \to \{1, \ldots, p\}$ and $f_2 : \{0,1\}^* \to \{0,1\}^\ell$. First, the above algorithm $\mathsf{Extract}(vk, sk)$ satisfies the second condition for the CPRed reduction given in Section 5.2. Second, by Zhandry's RO simulation technique (i.e., replacing the random functions (f_1, f_2) with k-wise independent functions) and [35, Corollary 4.3], one can check that the above $\mathsf{Extract}(vk, sk)$ also satisfies the first condition.

By Theorem 10, we have that there are reductions from the security of the FDH signature [4] and its variant the PSF-FDH signature, the GPV-IBE scheme [18], and the "implicit-rejection" KEM variants from TDP [30] and the FO transform [16,20], to their underlying hard problems in the QROM. We clarify that Theorem 10 provides guarantee on the security of a scheme only when its underlying problems are hard for quantum polynomial time adversaries. In particular, Theorem 10 implies that there is a reduction from the IND-ID-CPA security of the BF-IBE scheme [8] to the hardness of the BDH problem in the QROM, but it provides no guarantee on the security of BF-IBE scheme against quantum adversaries since the BDH problem can be efficiently solved by using the Shor algorithm [29]. We finally note that Theorem 10 not only subsumes the security reductions for the FDH signature [4], the PSF-FDH signature and the GPV-IBE scheme [18] given in [7,35], and for the "implicit rejections" KEMs from the FO transform [16,20] given in [22], but also gives new security reductions for the "implicit-rejection" KEM variants from TDP [30] in the QROM.

We now give a formal proof of Theorem 10, which are mainly based on the facts that 1) for a $(\{F_\lambda\}_{\lambda \in (0,1)}, I)$-CPRed, the simulation of the RO and that of the $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ are relatively independent; and that 2) the simulation of the RO can be "safely" quantized by the assumption in Theorem 10.

Proof. Let $\mathcal{A}$ be any quantum algorithm that solves problem Q with non-negligible advantage ϑ_A, by making at most q_1 queries to $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ and q_2 quantum RO (QRO) queries. Let $\lambda \in (0,1)$ be a parameter will be determined later. Let $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4)$ be the (F_λ, I)-splittable reduction in Definition 8. We first give the description of the reduction $\mathcal{S}'$.

Formally, given the security parameter κ and an instance x of P as inputs, $\mathcal{S}'$ first computes $(y, st_1 = (\tau, f)) \leftarrow \mathcal{S}_1^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}(1^\kappa, x)$ and sets $st_2 = st_3 = \epsilon$. Then, it uniformly chooses $k^* \xleftarrow{\$} \{1, \ldots, q_2\}$. Define the quantum oracle $\mathcal{O}_{F_\lambda}(\cdot) : |h_1, h_2\rangle \to |h_1, F_\lambda(\tau, h_1) \oplus h_2\rangle$ and $\mathcal{O}_f(\cdot) : |h_1, h_2\rangle \to |h_1, f(h_1) \oplus h_2\rangle$. Then, $\mathcal{S}'$ invokes $\mathcal{A}$ with input y, answers the $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ queries from $\mathcal{A}$ by running $st_2 \leftarrow \mathcal{S}_2^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}(y, st_1)$ and handles a quantum RO query $|\phi\rangle = |h_1, h_2\rangle$ from $\mathcal{A}$ as follows: if $|\phi\rangle$ is a Type-I RO query, return $|\psi\rangle = \mathcal{O}_{F_\lambda}(|\phi\rangle)$ to $\mathcal{A}$. Otherwise, if $|\phi\rangle$ is the k^*-th Type-II RO query, measure the argument of the query and set st_3 to be the measurement outcome, else return $\mathcal{O}_f(|\phi\rangle)$ to $\mathcal{A}$. If $\mathcal{S}_2^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}(y, st_1)$ aborts, then $\mathcal{S}'$ runs a trivial algorithm for problem P and outputs whatever it obtains. Otherwise, $\mathcal{S}'$ computes and outputs $w \leftarrow \mathcal{S}_4^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}(st_1, st_2, st_3)$ as the solution to its own challenge x of problem P.

It suffices to show that $\mathcal{S}'$ is a valid reduction in the QROM. We will finish the proof by using a sequence of six main games G_i for $i \in \{0, \ldots, 5\}$, where game G_0 is the real game and game G_5 is essentially the game simulated by $\mathcal{S}'$. For technical reason, we also define a sequence of sub-games $G_{3,j}$ between the two main games G_3 and G_4 for $j = \{0, \ldots, dq_1\}$ such that $G_{3,0} = G_3$, $G_{3,dq_1} = G_4$, and $G_{3,j}$ is modified from $G_{3,j-1}$ by replacing at most one Type-II RO response with a uniformly random one (to apply Lemma 1). We outline the changes between two consecutive games in Table 1. The game sequences is formally given below.

Game G_0: This game is the real security game of Q in the QROM. Concretely, the challenger $\mathcal{C}_\mathrm{Q}$ picks two functions $f_1 \xleftarrow{\$} \mathcal{F}_{n,m}$ and $f_2 \xleftarrow{\$} \mathcal{F}_{n,m}$ for Type-I and Type-II RO queries, respectively. Set $\mathcal{O}_{f_1}(\cdot) : |h_1, h_2\rangle \to$

Table 1. Outline of the changes between two consecutive games

Games	Changes w.r.t. Previous Game	Note
G_0	**Instance:** pick a $f_1, f_2 \xleftarrow{\$} \mathcal{F}_{n,m}$, and set $\mathcal{O}_{f_1}(\cdot): \lvert h_1, h_2\rangle \to \lvert h_1, f_1(h_1) \oplus h_2\rangle$ $\mathcal{O}_{f_2}(\cdot): \lvert h_1, h_2\rangle \to \lvert h_1, f_2(h_1) \oplus h_2\rangle$ compute $(y, s_y) \leftarrow \mathsf{Q.IGen}^{\mathcal{O}(\cdot)}(1^\kappa)$ Q.Orcl **query** z: compute $u = \mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, z)$ **QRO query** $\lvert\phi\rangle$: compute $\mathcal{O}_{quant}(\phi)$	real game
G_1	compute $(x, s_x, st_1 = (\tau, f)) \leftarrow \mathsf{Q.Extract}(y, s_y)$. After $\mathcal{A}$ outputs a solution and terminates, simulate the behavior of $\mathcal{A}$ to $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$, and omits the output of $\mathcal{A}$ if $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ aborts during the simulation	a conceptual change of G_0 in the adversary's view
G_2	answer the RO queries using $\mathcal{O}_{F_\lambda}(\cdot): \lvert h_1, h_2\rangle \to \lvert h_1, F_\lambda(\tau, h_1) \oplus h_2\rangle$ and $\mathcal{O}_f(\cdot): \lvert h_1, h_2\rangle \to \lvert h_1, f(h_1) \oplus h_2\rangle$ for Type I and Type II RO queries, respectively (i.e., replace $\mathcal{O}_{f_1}(\cdot)$ and $\mathcal{O}_{f_2}(\cdot)$ in game G_1 with $\mathcal{O}_{F_\lambda}(\cdot)$ and $\mathcal{O}_f(\cdot)$, respectively)	the difference on the outputs between G_2 and G_1 is bounded by $\mathsf{poly}_2'(q_3)\lambda^2$
G_3	for each $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z, immediately feed z to $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$, and abort if $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ aborts	a conceptual change of G_2
$G_{3,j}$	let $k_j = \lfloor \frac{j-1}{d} \rfloor + 1$ and $\ell_j = j \mod d$. For the ℓ_j-th RO query h_{k_j,ℓ_j} in computing the response to the k_j-th $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z_{k_j} and $\ell_j \in I(z_{k_j})$, namely, h_{k_j,ℓ_j} is a "bad" Type-II RO query, replace $f(h_{k_j,\ell_j})$ with a random and consistent r_{k_j,ℓ_j}	let $G_{3,0} = G_3$, then games $G_{3,j-1}$ and $G_{3,j}$ are connected by the One-way to Hiding Lemma 1
G_4	for each $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z, replace $f(h_\ell)$ with a random and consistent r_ℓ if h_ℓ is ℓ-th RO query in computing the response to z and $\ell \in I(z)$, namely, h_ℓ is a "bad" Type-II RO query	a conceptual change of G_{3,dq_1}
G_5	compute $(x, s_x) \leftarrow \mathrm{P.IGen}(1^\kappa), (y, st_1) \leftarrow \mathcal{S}_1^{\mathrm{P.Orcl}(s_x,\cdot)}(x)$, and use $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ to answer all $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ queries	the responses to $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ queries generated by $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ is statistically close to that generated in G_4

$\lvert h_1, f_1(h_1) \oplus h_2\rangle$ and $\mathcal{O}_{f_2}(\cdot): \lvert h_1, h_2\rangle \to \lvert h_1, f_2(h_1) \oplus h_2\rangle$. Then, computes $(y, s_y) \leftarrow \mathsf{Q.IGen}^{\mathcal{O}(\cdot)}(1^\kappa)$, and invokes $\mathcal{A}$ with input y:

- When receiving a QRO query $\lvert\phi\rangle = \lvert h_1, h_2\rangle$ from $\mathcal{A}$, return $\mathcal{O}_{f_1}(\lvert\phi\rangle)$ to $\mathcal{A}$ if this is a Type-I RO query, else return $\mathcal{O}_{f_2}(\lvert\phi\rangle)$ if this is a Type-II RO query;
- When receiving a $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z from $\mathcal{A}$, compute $u = \mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, z)$ by using s_y. Whenever the computation stops to make a RO query h, use $f_1(h)$ if h is a Type-I RO query or $f_2(h)$ if h is a Type-II RO query to continue the computation. Finally, return u to $\mathcal{A}$.
- When $\mathcal{A}$ outputs a solution v of y, compute and output $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v)$.

Let Succ_i be the event that the the challenger outputs 1, i.e., the algorithm $\mathcal{A}(y)$ succeeds to output a solution v of y such that $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v) = 1$, where $i \in \{0, \ldots, 5\}$. By definition, the advantage of $\mathcal{A}$ in game G_0 is equal to $\Pr[\mathsf{Succ}_0] - t(\kappa)$, i.e., $\Pr[\mathsf{Succ}_0] = \vartheta_A + t(\kappa)$, where $t(\cdot)$ is the threshold function for problem Q in Definition 6.

Game G_1: This game is almost identical to game G_0 except the following: the challenger $\mathcal{C}_{\mathrm{Q}}$ records all the $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ queries from $\mathcal{A}$. After $\mathcal{A}$ finishes and outputs v, $\mathcal{C}_{\mathrm{Q}}$ computes $(x, s_x, st_1) \leftarrow \mathsf{Q.Extract}(y, s_y)$, and simulate the behavior of $\mathcal{A}$ to $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$. If $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ aborts during the simulation, $\mathcal{C}_{\mathrm{Q}}$ runs a trivial solving algorithm to find a solution v' of y and outputs $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v')$ (In this case, we have $\Pr\left[\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v') = 1\right] = t(\kappa) + \mathrm{negl}(\kappa)$. Otherwise, $\mathcal{C}_{\mathrm{Q}}$ outputs $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v)$.

Let E_1 be the event that $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}$ aborts in game G_1.

Lemma 4. $\Pr[\mathsf{Succ}_1] \geq t(\kappa) + \lambda\vartheta_A - \mathrm{poly}_1(q_1)\lambda^2 + \mathrm{negl}(\kappa)$, *where* $\mathrm{poly}_1(\cdot)$ *is the fixed polynomial in Definition 8.*

Proof. Note that the view of $\mathcal{A}$ in both games G_0 and G_1 are identical. Thus, if E_1 does not happen, the probability that $\mathcal{C}_{\mathrm{Q}}$ outputs 1 in game G_1 is equal to that in game G_0 (i.e., $\Pr[\mathsf{Succ}_1|\neg E_1] = \Pr[\mathsf{Succ}_0]$), else the probability that $\mathcal{C}_{\mathrm{Q}}$ outputs 1 is $t(\kappa) + \mathrm{negl}(\kappa)$ (i.e., $\Pr[\mathsf{Succ}_1|E_1] = t(\kappa) + \mathrm{negl}(\kappa)$). By the law of total probability, we have that $\Pr[\mathsf{Succ}_1] = \vartheta_A \cdot \Pr[\neg E_1] + t(\kappa) + \mathrm{negl}(\kappa)$. In addition, by Definition 8 the probability $\Pr[\neg E_1] \geq \lambda - \mathrm{poly}_1(q_1)\lambda^2$ for any even unbounded algorithm $\mathcal{A}$. Since $0 \leq \vartheta_A \leq 1$, we have that $\Pr[\mathsf{Succ}_1] \geq t(\kappa) + \lambda\vartheta_A - \mathrm{poly}_1(q_1)\lambda^2 + \mathrm{negl}(\kappa)$. This completes the proof. □

Game G_2: This game is almost identical to game G_1 except the following: after obtaining (y, s_y), $\mathcal{C}_{\mathrm{Q}}$ computes $(x, s_x, st_1 = (\tau, f)) \leftarrow \mathrm{Q}.\mathsf{Extract}(y, s_y)$. Then, it answers all the Type-I and Type-II RO queries using $F_\lambda(st_1, \cdot)$ and $f(\cdot)$ for classical queries from $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ and $\mathcal{O}_{F_\lambda}(\cdot) : |h_1, h_2\rangle \to |h_1, F_\lambda(st_1, h_1) \oplus h_2\rangle$ and $\mathcal{O}_f(\cdot) : |h_1, h_2\rangle \to |h_1, f(h_1) \oplus h_2\rangle$ for quantum RO queries from $\mathcal{A}$.

Lemma 5. *There exist is a fixed polynomial* $\mathsf{poly}_3(\cdot,\cdot)$ *such that for the choice of* $\lambda = 2\vartheta_A/\mathrm{poly}_3(q_1, q_3)$*, we have* $\Pr[\mathsf{Succ}_2] \geq t(\kappa) + \vartheta_A^2/\mathsf{poly}_3(q_1, q_3) + \mathrm{negl}(\kappa)$.

Proof. As f is real random function, the oracle $\mathcal{O}_f(\cdot)$ in game G_2 is identical to $\mathcal{O}_{f_2}(\cdot)$ in game G_1. Thus, the only difference between games G_2 and G_1 is that $\mathcal{C}_{\mathrm{Q}}$ replaces the real RO $\mathcal{O}_{f_1}(\cdot)$ in game G_1 with $\mathcal{O}_{F_\lambda}(\cdot)$ in game G_2. Since $\mathcal{A}$ will make at most q_1 $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ queries (each $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ query requires at most d RO queries) and q_2 quantum RO queries, the output distribution of $\mathcal{A}$ in game G_2 is at most $\mathrm{poly}'_2(q_3)\lambda^2$ to that in game G_1 by assumption, where $q_3 \geq dq_1 + q_2$. Thus, $\Pr[\mathsf{Succ}_2] \geq \Pr[\mathsf{Succ}_1] - \mathsf{poly}'_2(q_3)\lambda^2$. As $\Pr[\mathsf{Succ}_1] \geq t(\kappa) + \lambda\vartheta_A - \mathrm{poly}_1(q_1)\lambda^2 + \mathrm{negl}(\kappa)$, by setting $\mathrm{poly}_3(q_1, q_3) = 4(\mathrm{poly}_1(q_1) + \mathrm{poly}'_2(q_3))$ and $\lambda = 2\vartheta_A/\mathrm{poly}_3(q_1, q_3)$, we have $\Pr[\mathsf{Succ}_2] \geq t(\kappa) + \lambda\vartheta_A - (\mathrm{poly}_1(q_1) + \mathsf{poly}'_2(q_3))\lambda^2 + \mathrm{negl}(\kappa) \geq t(\kappa) + \vartheta_A^2/\mathsf{poly}_3(q_1, q_3) + \mathrm{negl}(\kappa)$, which completes the proof. □

Game G_3: This game is almost identical to game G_2 except that whenever receiving a $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ query z from $\mathcal{A}$, $\mathcal{C}_{\mathrm{Q}}$ first simulates the response to z by using $\mathcal{S}_2^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}(y, st_1)$. If $\mathcal{S}_2^{\mathrm{P}.\mathsf{Orcl}(s_x,\cdot)}(y, st_1)$ aborts, $\mathcal{C}_{\mathrm{Q}}$ runs a trivial solving algorithm to find a solution v' of y, outputs $\mathrm{Q}.\mathsf{Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v')$ and terminates. Otherwise, $\mathcal{C}_{\mathrm{Q}}$ performs as in game G_2.

Lemma 6. $\Pr[\mathsf{Succ}_3] = \Pr[\mathsf{Succ}_2]$.

Proof. This lemma follows because game G_3 is a conceptual change of G_2. □

Let $G_{3,0} = G_3$ and $\mathsf{Succ}_{3,0} = \mathsf{Succ}_3$. For $1 \leq j \leq dq_1$, define $k_j = \lfloor \frac{j-1}{d} \rfloor + 1$ and $\ell_j = j \mod d$. Now, we define a sequence of sub-games $G_{3,j}$ for $1 \leq j \leq dq_1$. Informally, game $G_{3,j}$ is identical to game $G_{3,j-1}$ except that when computing the response to the k_j-th $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ query z_{k_j} from $\mathcal{A}$, the challenger $\mathcal{C}_{\mathrm{Q}}$ replaces $f(h_{k_j,\ell_j})$ in game $G_{3,j-1}$ with a random and consistent r_{k_j,ℓ_j} (chosen from an appropriate domain) in game $G_{3,j}$ if h_{k_j,ℓ_j} is the ℓ_j-th RO query required by $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ in computing the response to z_{k_j} and $\ell_j \in I(z_{k_j}) \subseteq \{1, \ldots, d\}$. Clearly, $\mathcal{A}$ can only distinguish game $G_{3,j}$ from $G_{3,j-1}$ by making a QRO query containing h_{k_j,ℓ_j}. Let $\mathsf{Succ}_{3,j}$ be the event that $\mathcal{O}$ outputs 1 in game $G_{3,j}$, where $j \in \{1, \ldots, dq_1\}$. Let L be an initially empty list in the following games, which is used to keep consistency of the "bad" Type-II RO responses.

Game $G_{3,j}$: Let $k_j = \lfloor \frac{j-1}{d} \rfloor + 1$ and $\ell_j = j \mod d$. This game is almost identical to game $G_{3,j-1}$ except that the challenger $\mathcal{C}_{\mathrm{Q}}$ handles the ℓ_j-th RO query h_{k_j,ℓ_j} required in computing the response u_{k_j} to the k_j-th $\mathrm{Q}.\mathsf{Orcl}^{\mathcal{O}(\cdot)}$ query z_{k_j} from $\mathcal{A}$ as follows: if $\ell_j \notin I(z_{k_j})$, set the response to the RO query h_{k_j,ℓ_j} as in game $G_{3,j-1}$. Else, if there does not exist a pair $(h_{k_j,\ell_j}, r) \in L$ for some r, choose a uniformly random r as the response to the RO query h_{k_j,ℓ_j}, and append (h_{k_j,ℓ_j}, r) to the list L.

Let $\mathcal{B}_{3,j}$ be an algorithm which interacts with $\mathcal{A}$ as the challenger in game $G_{3,j}$ except that it chooses an integer $k^* \xleftarrow{\$} \{1, \ldots, q_2\}$, runs $\mathcal{A}$ until just after receiving the k^*-th QRO query $|\phi\rangle = |h_1, h_2\rangle$ from $\mathcal{A}$, measures the argument of the query in the computational basis, and outputs the measurement result $\hat{h}$ ($\mathcal{B}_{3,j}$

outputs $\perp$ if $\mathcal{A}$ makes less than k^* RO queries). Let $\delta_{3,j}$ be the probability that $\hat{h} = h_{k_j,\ell_j} \wedge \ell_j \in I(z_{k_j})$, where z_{k_j} is the k_j-th $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query from $\mathcal{A}$, and h_{k_j,ℓ_j} is the ℓ_j-th RO query required for computing $u = \mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, z_{k_j})$. Clearly, $\delta_{1,j}$ is only useful for a (F_λ, I)-CPRed with $I(\cdot) \neq \emptyset$.

Lemma 7. *If $I(\cdot) = \emptyset$, then $\Pr[\mathsf{Succ}_{3,j-1}] = \Pr[\mathsf{Succ}_{3,j}]$, else we have that $|\Pr[\mathsf{Succ}_{3,j-1}] - \Pr[\mathsf{Succ}_{3,j}]| \leq 2q_2\sqrt{\delta_{3,j}}$ holds.*

Proof. Note that if $I(\cdot) = \emptyset$, the change from game $G_{3,j-1}$ to game $G_{3,j}$ is conceptual, i.e., we always have that $\Pr[\mathsf{Succ}_{3,j-1}] = \Pr[\mathsf{Succ}_{3,j}]$. Otherwise, the only difference between $G_{3,j-1}$ and $G_{3,j}$ is that the challenger $\mathcal{C}_{\mathrm{Q}}$ might replace the value $f(h_{k_j,\ell_j})$ in game $G_{3,j-1}$ with a uniformly random r_{k_j,ℓ_j} in game $G_{3,j}$ if z_{k_j} is the k_j-th $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query from $\mathcal{A}$, h_{k_j,ℓ_j} is the ℓ_j-th RO query required for computing $u_{k_j} = \mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, z_{k_j})$ and $\ell_j \in I(z_{k_j})$. By using the fact that f is a random function and Lemma 1, we have that $|\Pr[\mathsf{Succ}_{3,j-1}] - \Pr[\mathsf{Succ}_{3,j}]| \leq 2q_2\sqrt{\delta_{3,j}}$ holds. This completes the proof. □

Game G_4: The challenger $\mathcal{C}_{\mathrm{Q}}$ first computes $(y, s_y) \leftarrow \mathsf{Q.IGen}^{\mathcal{O}(\cdot)}(1^\kappa)$, and $(x, s_x, st_1 = (\tau, f)) \leftarrow \mathsf{Q.Extract}(y, s_y)$. Let $\mathcal{O}_f(\cdot) : |h_1, h_2\rangle \to |h_1, f(h_1) \oplus h_2\rangle$ and $\mathcal{O}_{F_\lambda}(\cdot) : |h_1, h_2\rangle \to |h_1, F_\lambda(st_1, h_1) \oplus h_2\rangle$. Then, $\mathcal{C}_{\mathrm{Q}}$ invokes $\mathcal{A}$ with y:

- When receiving a quantum RO query $|\phi\rangle = |h_1, h_2\rangle$ from $\mathcal{A}$, return $\mathcal{O}_{F_\lambda}(|\phi\rangle)$ to $\mathcal{A}$ if $|\phi\rangle$ is a Type-I RO query, other return $\mathcal{O}_f(|\phi\rangle)$ to $\mathcal{A}$ if it is a Type-II RO query;
- When receiving a $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z_i from $\mathcal{A}$, feed z to $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$. If $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ aborts, run a trivial solving algorithm to find a solution v' of y, output $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v')$ and terminate. Else, compute $u = \mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, z)$ by using s_y and count the RO queries required by the computation. Whenever the computation stops to make the ℓ-th RO query h for some $\ell \in \{1, \ldots, d\}$, distinguish the following cases:
  - If h is a Type-I RO query, use $F_\lambda(st_1, h)$ to continue the computation.
  - Else if h is a Type-II RO query and $\ell \notin I(z)$, use $f(h)$ to continue the computation. Otherwise, if there does not exist a pair $(h, r) \in L$, choose a random r to continue the computation and append (h, r) to the list L.

  Finally, return u to $\mathcal{A}$;
- Whenever $\mathcal{A}$ outputs a solution v of y, $\mathcal{C}_{\mathrm{Q}}$ feed v to $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$. If $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ aborts, run a trivial solving algorithm to find a solution v' of y and output $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v')$. Otherwise, output $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v)$.

Let $\mathcal{B}_i$ be an algorithm which interacts with $\mathcal{A}$ as the challenger in game G_i for $i \in \{4, 5, 6\}$ except that it chooses an integer $k^* \xleftarrow{\$} \{1, \ldots, q_2\}$, runs $\mathcal{A}$ until just after receiving the k^*-th QRO query $|\phi\rangle = |h_1, h_2\rangle$ from $\mathcal{A}$, measures the argument of the query and outputs the measurement outcome $\hat{h}$ ($\mathcal{B}_i$ outputs $\perp$ if $\mathcal{A}$ makes less than k^* RO queries). Let δ_i be the probability that $\hat{h} = h_{k_j,\ell_j} \wedge \ell_j \in I(z_{k_j})$ for any $1 \leq j \leq dq_1$, where z_{k_j} is the k_j-th $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query from $\mathcal{A}$, and h_{k_j,ℓ_j} is the ℓ_j-th RO query required for computing $u_{k_j} = \mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}(s_y, z_{k_j})$.

Lemma 8. $\Pr[\mathsf{Succ}_4] = \Pr[\mathsf{Succ}_{3,dq_1}]$. *Moreover, for $I(\cdot) \neq \emptyset$, we have $\delta_4 \geq \delta_{3,j}$ for any $1 \leq j \leq dq_1$.*

Proof. This first claim follows from the fact that games G_4 and G_{3,dq_1} are identical. Note that if $\mathcal{A}$ does not make a QRO query containing any element in $\{\{h_{k_j,\ell_j}\}_{\ell_j \in I(z_{k_j})}\}_{1 \leq j \leq dq_1}$, the view of $\mathcal{A}$ in games $G_{3,j}$ for $0 \leq j \leq dq_1$ is identically distributed. Moreover, before $\mathcal{A}$ making a QRO query containing one of the elements in $\{\{h_{k_j,\ell_j}\}_{\ell_j \in I(z_{k_j})}\}_{1 \leq j \leq dq_1}$, we always have that the view of $\mathcal{A}$ in games $G_{3,j}$ for $0 \leq j \leq dq_1$ is identically distributed, which implies that $\delta_4 \geq \delta_{3,j}$ for any $1 \leq j \leq dq_1$. This completes the proof. □

Game G_5: This game is almost identical to game G_4 except that $\mathcal{C}_{\mathrm{Q}}$ generates (x, s_x, y, st_1) by running $(x, s_x) \leftarrow \mathsf{P.IGen}(1^\kappa), (y, st_1) \leftarrow \mathcal{S}_1^{\mathrm{P.Orcl}(s_x,\cdot)}(x)$, and handles the $\mathsf{Q.Orcl}^{\mathcal{O}(\cdot)}$ query by directly using $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$. Whenever $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ aborts, $\mathcal{C}_{\mathrm{Q}}$ first runs a trivial solving algorithm to find a solution v' of y, then outputs $\mathsf{Q.Vrfy}^{\mathcal{O}(\cdot)}(y, s_y, v')$ and terminates.

Lemma 9. $|\Pr[\mathsf{Succ}_5] - \Pr[\mathsf{Succ}_4]| \leq \mathrm{negl}(\kappa)$. *Moreover, for $I(\cdot) \neq \emptyset$, we have $|\delta_5 - \delta_4| \leq \mathrm{negl}(\kappa)$.*

Proof. There are only two differences between game G_5 and game G_6: the way of generating (x, s_x, y, st_1) and the way of answering the $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ queries. First, by the assumption on $\mathsf{Q.Extract}$, we have that the distribution of (x, s_x, y, st_1) generated by using $(x, s_x) \leftarrow \mathrm{P.IGen}(1^\kappa)$ and $(y, st_1) \leftarrow \mathcal{S}_1^{\mathrm{P.Orcl}(s_x,\cdot)}(1^\kappa, x)$ in game G_5 is identical to that generated by using $(y, s_y) \leftarrow \mathrm{Q.IGen}^{\mathcal{O}(\cdot)}(1^\kappa)$ and $(x, s_x, st_1) \leftarrow \mathsf{Q.Extract}(y, s_y)$ in game G_4. Second, for each $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z, by the definition of game G_4, except at the RO queries with positions specified by $I(z)$, the behavior of generating the response to z in game G_4 is identical to that of the real $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ with the RO $\mathcal{O}(\cdot)$ being changed to the simulated RO by $\mathcal{S}_3$. Moreover, by the third condition in Definition 8 that for each $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ query z, except at the RO queries with positions specified by $I(z)$, the behavior of $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ in game G_5 is statistically close to that of the real $\mathrm{Q.Orcl}^{\mathcal{O}(\cdot)}$ with the RO $\mathcal{O}(\cdot)$ being changed to the simulated RO by $\mathcal{S}_3$. This means that except at the RO queries with positions specified by $I(z)$, the behavior of generating the response to z in game G_4 is statistically close to that of $\mathcal{S}_2^{\mathrm{P.Orcl}(s_x,\cdot)}(y, st_1)$ in game G_5. As the behaviors of both games G_4 and G_5 are identical at the RO queries with positions specified by $I(z)$, we have that game G_5 and G_4 are statistically indistinguishable. This completes the proof. □

It is easy to check that the view of $\mathcal{A}$ in game G_5 is identical to the one simulated by $\mathcal{S}'$. Thus, it suffices to analyze the behavior of $\mathcal{A}$ in game G_5. First, if $I(\cdot) = \emptyset$, by Lemma 4∼9 we have $\Pr[\mathsf{Succ}_5] \geq t(\kappa) + \vartheta_A^2/\mathsf{poly}_3(q_1, q_3) + \mathrm{negl}(\kappa)$. This means that $\mathcal{S}'$ can find a solution v of y with advantage at least $\vartheta_A^2/\mathsf{poly}_3(q_1, q_3)$ from $\mathcal{A}$. By computing $w \leftarrow \mathcal{S}_4^{\mathrm{P.Orcl}(s_x,\cdot)}(st_1, v)$, we have that $\mathcal{S}'$ can output a solution w of x with advantage $\vartheta_A^2/\mathsf{poly}_3(q_1, q_3)$ by Definition 8.

Second, if $I(\cdot) \neq \emptyset$, by Definition 8 the advantage that $\mathcal{A}$ outputs a valid solution v of y in game G_5 is negligible (i.e., $\Pr[\mathsf{Succ}_5] = \mathrm{negl}(\kappa)$). By Lemma 4∼9 we have $t(\kappa) + \vartheta_A^2/\mathsf{poly}_3(q_1, q_3) + \mathrm{negl}(\kappa) \leq \Pr[\mathsf{Succ}_2] \leq \Pr[\mathsf{Succ}_5] + 2q_2 \sum_{j=1}^{dq_1} \sqrt{\delta_{3,j}} + \mathrm{negl}(\kappa) \leq t(\kappa) + 2dq_1q_2\sqrt{\delta_5} + \mathrm{negl}(\kappa)$. Thus, we have that

$$\vartheta_A^2/\mathsf{poly}_3(q_1, q_3) \leq 2dq_1q_2\sqrt{\delta_5} + \mathrm{negl}(\kappa).$$

Thus, we have that δ_5 is non-negligible if ϑ_A is non-negligible. Note that $\mathcal{S}'$ will compute $w \leftarrow \mathcal{S}_4^{\mathrm{P.Orcl}(s_x,\cdot)}(st_1, \hat{h})$ by first randomly measuring one of the QRO queries from $\mathcal{A}$ to obtain $\hat{h}$. The probability that $\hat{h}$ is a "bad" RO query is equal to δ_5 by definition, which is non-negligible. Thus, we have that $\mathcal{S}'$ can output a solution w of x with non-negligible advantage by Definition 8. This completes the proof. □

7 Incomparability of CPReds with NPReds and RPReds

By restricting the programmability, Fischlin et al. [14] formalized three notions of BB-reductions in the ROM: fully-programming reduction (FPRed), randomly-programming reduction (RPRed) and non-programming reduction (NPRed). In brief, the FPRed formalizes the standard conception of BB-reductions in the ROM which have full control over the RO, while the NPRed considers the setting where all parties are given access to an external RO (i.e., non-programmable RO [26]) that is not controlled by any party (but the reduction is allowed to learn all the RO queries made by $\mathcal{A}$). The RPRed can only program the RO via an interface not fully controlled by itself, and is formalized via the notion of randomly programmable RO. Formally, a RPRO $\mathcal{O} = (R_{eval}, R_{rand}, R_{prog})$ is an idealized object consisting of a public interface R_{eval}, and two private interfaces R_{rand} and R_{prog}, where R_{eval} behaves as a conventional RO mapping $Dom \to Rng$; R_{rand} maps a string in $\{0,1\}^*$ to a random value in Rng; and $R_{prog}(X, Y)$ takes an $X \in Dom$ and a string $Y \in \{0,1\}^*$ as inputs, sets $R_{eval}(X) = R_{rand}(Y)$. As NPReds, a RPRed (i.e., a black-box reduction in the model equipped with a RPRO) is allowed to learn all the RO-queries made by the adversary $\mathcal{A}$ via the public interface R_{eval}, but unlike NPReds, a RPRed is also allowed to program the answer of a given RO-query X made by the adversary $\mathcal{A}$ via the two private interfaces R_{rand} and R_{prog}. Specifically, after receiving a RO query X from $\mathcal{A}$, the reduction can make a number of queries to R_{rand} and R_{prog} before returning the answer $R_{eval}(X)$ (which can be programmed by using the interface R_{prog}) to $\mathcal{A}$. Note that a RPRed cannot directly set the RO responses with any values of his own choices.

In this following, we investigate the relation of CPReds to the notions of BB-reductions formalized by Fischlin et al. [14], i.e., fully-programming reduction (FPRed), randomly-programming reduction (RPRed), and non-programming reduction (NPRed). As shown in [14], a NPRed implies a RPRed (i.e., NPRed $\Rightarrow$ RPRed), which in turn implies a FPRed by definition (i.e., RPRed $\Rightarrow$ FPRed). Since the notion of FPRed essentially formalizes the standard concept of all BB-reductions in the ROM, we immediately have that a CPRed implies a FPRed (i.e., CPRed $\Rightarrow$ FPRed). By carefully examining the OAEP encryption [3] and the FDH signature [4], we show that NPRed $\not\Rightarrow$ CPRed in Section 7.1 and CPRed $\not\Rightarrow$ RPRed in Section 7.2. Combining our results with the fact that NPRed $\Rightarrow$ RPRed, we immediately have that CPRed $\not\Rightarrow$ NPRed and RPRed $\not\Rightarrow$ CPRed, which implies that CPReds are incomparable to both NPReds and RPReds.

We begin by first recalling the definition of trapdoor permutations (TDP). A trapdoor permutation family $\Pi_{\mathrm{TDP}} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv})$ with respect to length function $\ell(\cdot)$ consists of three algorithms. The parameter generator $\mathsf{Gen}(1^\kappa)$ takes a security parameter κ as input, outputs an index-trapdoor pair (s, td), i.e., $(s, td) \leftarrow \mathsf{Gen}(1^\kappa)$. The evaluation algorithm $\mathsf{Eval}(s, x)$ takes an index s and a string $x \in \{0,1\}^{\ell(\kappa)}$ as inputs, outputs a string $y \in \{0,1\}^{\ell(\kappa)}$, i.e., $y \leftarrow \mathsf{Eval}(s, x)$. The inversion algorithm $\mathsf{Inv}(td, y)$ takes a trapdoor td and a string $y \in \{0,1\}^{\ell(\kappa)}$ as inputs, outputs a string $x \in \{0,1\}^{\ell(\kappa)}$, i.e., $x \leftarrow \mathsf{Inv}(td, y)$.

For correctness, we require that for all $(s, td) \leftarrow \mathsf{Gen}(1^\kappa)$ and all $x \in \{0,1\}^{\ell(\kappa)}$, the equation $\mathsf{Inv}(td, \mathsf{Eval}(s, x)) = x$ always holds. Moreover, we say that $\Pi_{\mathrm{TDP}} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv})$ is *one-way* if for all PPT algorithm $\mathcal{A}$, we have that

$$\Pr\Big[(s, td) \leftarrow \mathsf{Gen}(1^\kappa), x \xleftarrow{\$} \{0,1\}^{\ell(\kappa)}, x' \leftarrow \mathcal{A}(s, \mathsf{Eval}(s, x)) : x' = x\Big] = \mathrm{negl}(\kappa).$$

A trapdoor permutation $\Pi_{\mathrm{TDP}} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv})$ is said to be *partial-domain one-way* with respect to $(\ell(\cdot), \ell_0(\cdot))$ if for all PPT algorithm $\mathcal{A}$, we have that

$$\Pr\left[\begin{array}{c}(s, td) \leftarrow \mathsf{Gen}(1^\kappa), x_0 \xleftarrow{\$} \{0,1\}^{\ell_0(\kappa)}, x_1 \xleftarrow{\$} \{0,1\}^{\ell(\kappa)-\ell_0(\kappa)} \\ x_0' \leftarrow \mathcal{A}(s, \mathsf{Eval}(s, x_0\|x_1)) : x_0' = x_0\end{array}\right] = \mathrm{negl}(\kappa).$$

7.1 NPRed $\not\Rightarrow$ CPRed

In this subsection, we show that the OAEP encryption in [3] which was provably secure under NPReds is not provable under CPReds. Formally, let κ be the security parameter. Let $n_1, n_2, n_3 \geq \kappa$ be any positive integers. Let $\ell(\cdot)$ and $\ell_0(\cdot)$ be two arbitrary integer functions such that $\ell(\kappa) = \ell_0(\kappa) + n_3 = n_1 + n_2 + n_3$. Let $\Pi_{\mathrm{TDP}} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv})$ be a trapdoor partial-domain one-way permutation family with respect to functions $(\ell(\cdot), \ell_0(\cdot))$. Let $G : \{0,1\}^{n_3} \to \{0,1\}^{n_1+n_2}$ and $H : \{0,1\}^{n_1+n_2} \to \{0,1\}^{n_3}$ be two hash functions. The PKE scheme $\Pi_{\mathrm{OAEP}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ given in [3] works as follows:

- $\mathsf{KeyGen}(1^\kappa)$: take a security parameter κ as input, compute and output the public and secret key pair $(pk, sk) = (s, td) \leftarrow \Pi_{\mathrm{TDP}}.\mathsf{Gen}(1^\kappa)$;
- $\mathsf{Enc}(pk, \mu)$: take the public key $pk = s$ and a message $\mu \in \{0,1\}^{n_1}$ as inputs, first append n_2 zeros to μ and obtain $\hat{\mu} = \mu\|0_{n_2} \in \{0,1\}^{n_1+n_2}$. Then uniformly choose $r \xleftarrow{\$} \{0,1\}^{n_3}$, compute $x_0 = \hat{\mu} \oplus G(r)$, $x_1 = r \oplus H(x_0)$ and $y = \Pi_{\mathrm{TDP}}.\mathsf{Eval}(s, x_0\|x_1)$. Finally, output the ciphertext $c = y$;
- $\mathsf{Dec}(sk, c)$: take the secret key $sk = td$ and a ciphertext $c = y \in \{0,1\}^{\ell(\kappa)}$ as inputs, first compute $x_0\|x_1 = \Pi_{\mathrm{TDP}}.\mathsf{Inv}(td, y)$, $r = x_1 \oplus H(x_0)$ and $\hat{\mu} = x_0 \oplus G(r)$, where $x_0, \hat{\mu} \in \{0,1\}^{n_1+n_2}$ and $x_1 \in \{0,1\}^{n_3}$. Then, parse $\hat{\mu} = \mu\|\mu_0$ into $\mu \in \{0,1\}^{n_1}$ and $\mu_0 \in \{0,1\}^{n_2}$. If $\mu_0 \neq 0_{n_2}$, return $\perp$. Else return μ.

We have the following lemma which is implicit in [17,14].

Lemma 10. *Let $n_1, n_2, n_3 \geq \kappa$ be any positive integers. Let $(\ell(\cdot), \ell_0(\cdot))$ be two arbitrary integer functions such that $\ell(\kappa) = \ell_0(\kappa) + n_3 = n_1 + n_2 + n_3$. If the trapdoor permutation family Π_{TDP} is partial-domain one-way with respect to $(\ell(\cdot), \ell_0(\cdot))$, the PKE scheme Π_{OAEP} is provably CCA-secure under NPReds.*

As noted in [14], the reduction for the PKE scheme Π_{OAEP} given in [17] does not need to program the RO, and thus is a NPRed. Specifically, assuming that the trapdoor permutation family $\Pi_{\mathrm{TDP}} = (\mathsf{Gen}, , \mathsf{Eval}, \mathsf{Inv})$ is partial-domain one-way, Π_{PKE} is provably CCA-secure under NPReds (see Section E for the CCA-security of PKE).

We note that the reduction in [17] crucially relies on the knowledge of all RO query/response pairs made by the adversary to answer the decryption queries, which is not allowed for a CPRed (since the simulation of the decryption oracle is oblivious to the RO queries made by the adversary). Actually, we can show that the scheme Π_{OAEP} is not provable under CPReds.

Theorem 11. *Let $n_1, n_2, n_3 \geq \kappa$ be any positive integers. Let $(\ell(\cdot), \ell_0(\cdot))$ be any two integer functions such that $\ell(\kappa) = \ell_0(\kappa) + n_3 = n_1 + n_2 + n_3$. There is no CPRed from the CCA-security of Π_{OAEP} to the partial-domain one-wayness of the trapdoor permutation family Π_{TDP} with respect to $(\ell(\cdot), \ell_0(\cdot))$.*

We first give a sketch of the proof. Technically, we will use the two-oracle separation technique of Hsiao and Reyzin [21] by giving two oracles Λ and Ω such that 1) Λ is an ideal trapdoor permutation and Ω is a "breaking" oracle; 2) there is a PPT adversary $\mathcal{A}^{\Lambda,\Omega}$ which can break the CCA-security of the instantiation $\Pi^{\Lambda}_{\mathrm{OAEP}}$ of Π_{OAEP} using Λ; and 3) no PPT CPRed $\mathcal{S}$, given access to Λ and $\mathcal{A}^{\Lambda,\Omega}$ (i.e., $\mathcal{S}^{\Lambda,\mathcal{A}^{\Lambda,\Omega}}$), can break the partial-domain one-wayness of Λ. The basic idea is that $\mathcal{S}$ can only access the "breaking" oracle Ω by interacting with $\mathcal{A}$ which will only use Ω after he is convinced that Ω is "useless" to $\mathcal{S}$ (which needs $\mathcal{S}$ to invert the permutation and correctly answer a random decryption query without "seeing" the RO queries). In other words, the ability of $\mathcal{A}$ to access the breaking oracle Ω cannot be utilized by $\mathcal{S}$ to break the partial-domain one-wayness of Λ.

Proof. Let $\mathcal{E} = (\mathcal{E}_{\mathsf{Perm}}, \mathcal{E}_E, \mathcal{E}_{E^{-1}})$ be an oracle which is initialized with a random permutation $\mathsf{Perm} : \{0,1\}^\kappa \to \{0,1\}^\kappa$ and a keyed family of random permutations $E : \{0,1\}^\kappa \times \{0,1\}^{\ell(\kappa)} \to \{0,1\}^{\ell(\kappa)}$ (i.e., given a key $s \in \{0,1\}^\kappa$, $E(s,\cdot) : \{0,1\}^{\ell(\kappa)} \to \{0,1\}^{\ell(\kappa)}$ is a random permutation). The two interfaces $\mathcal{E}_{\mathsf{Perm}}$ and $\mathcal{E}_E$ allow to evaluate Perm and E, respectively, whereas the interface $\mathcal{E}_{E^{-1}}$ allows to evaluate the inversion E^{-1} of E, i.e., $E^{-1}(s, E(s,x)) = x$. Now, we define the first oracle $\Lambda = (\mathsf{Gen}^{\mathcal{E}}, \mathsf{Eval}^{\mathcal{E}}, \mathsf{Inv}^{\mathcal{E}})$ consisting of three interfaces:

- $\mathsf{Gen}^{\mathcal{E}}(1^\kappa)$: first randomly choose $td \xleftarrow{\$} \{0,1\}^\kappa$, then send td to $\mathcal{E}_{\mathsf{Perm}}$ and obtain a response $s = \mathsf{Perm}(td)$. Finally, return (s, td), i.e., $(s, td) \leftarrow \mathsf{Gen}^{\mathcal{E}}(1^\kappa)$;
- $\mathsf{Eval}^{\mathcal{E}}(s,x)$: first send $(s,x) \in \{0,1\}^\kappa \times \{0,1\}^{\ell(\kappa)}$ to $\mathcal{E}_E$ and obtain a response $y = E(s,x)$. Then, return $y \in \{0,1\}^{\ell(\kappa)}$, i.e., $y \leftarrow \mathsf{Eval}^{\mathcal{E}}(s,x)$;
- $\mathsf{Inv}^{\mathcal{E}}(td,y)$: first send $td \in \{0,1\}^\kappa$ to $\mathcal{E}_{\mathsf{Perm}}$ and obtain a response $s = \mathsf{Perm}(td)$. Then, send $(s,y) \in \{0,1\}^\kappa \times \{0,1\}^{\ell(\kappa)}$ to $\mathcal{E}_{E^{-1}}$ and obtain a response $x = E^{-1}(s,y)$. Finally, return $x \in \{0,1\}^{\ell(\kappa)}$, i.e., $x \leftarrow \mathsf{Inv}^{\mathcal{E}}(td,y)$.

We first show that Λ is a family of ideal trapdoor permutations, i.e., no PPT algorithm $\mathcal{S}^\Lambda$ given oracle access to Λ can break the partial-domain one-wayness of Λ with respect to functions $\ell(\kappa) \geq \ell_0(\kappa) \geq \kappa$. Note that given a challenge pair $s^* = \mathsf{Perm}(td^*)$ and $y^* = E(s^*, x^*)$ as inputs, $\mathcal{S}^\Lambda$ is asked to output the first $\ell_0(\kappa)$-bit of x^* by making at most a polynomial number of queries to Λ. First, since Perm is a random permutation and $td^* \in \{0,1\}^\kappa$ is uniformly chosen at random, the probability that $\mathcal{S}^\Lambda$ can find td^* is negligible, which means that the inversion oracle $\Lambda.\mathsf{Inv}^{\mathcal{E}}$ cannot help $\mathcal{S}^\Lambda$ to invert $E(s^*,\cdot)$. Second, since $E(s^*,\cdot)$ is a random permutation, conditioned on that $\mathcal{S}^\Lambda$ cannot find td^*, the probability that $\mathcal{S}^\Lambda$ can output the first $\ell_0(\kappa)$-bit of x^* is also negligible (because it can only determine the first $\ell_0(\kappa)$-bit of x^* by making a query (s^*, x^*) to $\Lambda.\mathsf{Eval}^{\mathcal{E}}(\cdot,\cdot)$). This shows that Λ is partial-domain one-way.

The second oracle $\Omega = (\mathcal{R}_1, \mathcal{R}_2, \mathsf{Inv}^{\mathcal{E}})$ also consists of three interfaces, where $\mathcal{R}_1(\cdot)$ evaluates a random function from $\{0,1\}^{2^\kappa}$ to $\{0,1\}^{n_1}$, $\mathcal{R}_2(\cdot)$ evaluates a random function from $\{0,1\}^{2^\kappa}$ to $\{0,1\}^{n_3}$, and $\mathsf{Inv}^{\mathcal{E}}(\cdot,\cdot)$ simply relays its query to the oracle $\mathcal{E}$ via the interface $\mathcal{E}_{E^{-1}}$ and returns whatever it obtains from the oracle. Now, we give an adversary $\mathcal{A}^{\Lambda,\Omega}$ breaking the CCA-security of the Π^{Λ}_{PKE}, but no CPRed $\mathcal{S}^{\Lambda,\mathcal{A}^{\Lambda,\Omega}}$ can break the partial-domain one-wayness of Λ.

Description of $\mathcal{A}^{\Lambda,\Omega}$. Given a security parameter κ and a public key $pk = s$ of $\Pi^{\Lambda}_{\mathrm{OAEP}}$ as inputs, the goal of $\mathcal{A}^{\Lambda,\Omega}$ is to break the CCA-security of $\Pi^{\Lambda}_{\mathrm{OAEP}}$. The adversary $\mathcal{A}^{\Lambda,\Omega}$ which is also given access to an G-oracle and H-oracle used by the scheme $\Pi^{\Lambda}_{\mathrm{OAEP}}$ works as follows:

1. Let $V = \kappa \| s$ (i.e., the initial view of $\mathcal{A}^{\Lambda,\Omega}$);

2. Send $r = \Omega.\mathcal{R}_2(V)$ to the G-oracle and obtain a response z_1;
3. Update $V := V\|z_1$, and compute $\mu = \Omega.\mathcal{R}_1(V) \in \{0,1\}^{n_1}$;
4. Send $x_0 = (\mu\|0_{n_2}) \oplus z_1$ to the H-oracle and obtain a response z_2;
5. Send $y = \Lambda.\mathsf{Eval}^{\mathcal{E}}(s, x_0\|x_1))$ to the decryption oracle and obtain a result μ', where $x_1 = r \oplus z_2$;
6. If $\mu \neq \mu'$, output $\perp$ and abort;
7. Update $V := V\|z_2\|\mu'$, compute $\mu_0^* = \Omega.\mathcal{R}_1(V\|0), \mu_1^* = \Omega.\mathcal{R}_1(V\|1)$, and send (μ_0^*, μ_1^*) to obtain a challenge ciphertext y^*;
8. Compute $x^* = x_0^*\|x_1^* = \Omega.\mathsf{Inv}^{\mathcal{E}}(s, y^*)$, send $x_0^* \in \{0,1\}^{n_1+n_2}$ to the H-oracle and obtain a response z_2^*;
9. Send $r^* = x_1^* \oplus z_2^* \in \{0,1\}^{n_3}$ to the G-oracle and obtain a response z_1^*;
10. If $z_1^* \oplus x_0^* = \mu_0^*\|0_{n_2}$, output 0, else output 1.

Note that no matter how the oracles G and H are instantiated, $\mathcal{A}^{\Lambda,\Omega}$ can always break the CCA-security of $\Pi_{\mathrm{OAEP}}^{\Lambda}$. Let (s^*, y^*) be the challenge pair of the partial-domain one-wayness of Λ. We now show that no CPRed $\mathcal{S}(s^*, y^*)$ given oracle access to Λ and $\mathcal{A}^{\Lambda,\Omega}$ can find the first $\ell_0(\kappa)$-bit of $x^* = E^{-1}(s^*, y^*)$ with non-negligible probability. The basic idea is that $\mathcal{A}^{\Lambda,\Omega}$ will not use the inversion oracle $E^{-1}(s^*, \cdot)$ until it has been convinced that $\mathcal{S}^{\Lambda}$ can invert the function $E(s^*, \cdot)$, which is infeasible for $\mathcal{S}^{\Lambda}$ given only oracle access to Λ.

By Definition 8, a CPRed $\mathcal{S}^{\Lambda} = (\mathcal{S}_1^{\Lambda}, \mathcal{S}_2^{\Lambda}, \mathcal{S}_3^{\Lambda}, \mathcal{S}_4^{\Lambda})$ is a single-instance reduction and will not rewind the adversary $\mathcal{A}^{\Lambda,\Omega}$. Moreover, given a security parameter κ and a challenge pair (s^*, y^*) as inputs, $\mathcal{S}^{\Lambda} = (\mathcal{S}_1^{\Lambda}, \mathcal{S}_2^{\Lambda}, \mathcal{S}_3^{\Lambda}, \mathcal{S}_4^{\Lambda})$ works as follows:[10] First, it runs $(pk, st_1) \leftarrow \mathcal{S}_1^{\Lambda}(1^{\kappa}, (s^*, y^*))$ to obtain a public key pk and a state st_1. Second, it runs $st_2 \leftarrow \mathcal{S}_2^{\Lambda}(pk, st_1)$ to obtain a state st_2 by invoking the adversary $\mathcal{A}^{\Lambda,\Omega}$ and answering the decryption queries from $\mathcal{A}^{\Lambda,\Omega}$. Third, it runs $st_3 \leftarrow \mathcal{S}_3^{\Lambda}(st_1)$ to simulate the ROs for G, H and obtain a state st_3 which is either an empty string ϵ or one of the oracle queries to G and H. Finally, it runs $\mathcal{S}_4^{\Lambda}(st_1, st_2, st_3)$ to obtain a candidate solution x'.

Note that if $\mathcal{A}^{\Lambda,\Omega}$ does not use s^* as the first input in a query to $\Omega.\mathsf{Inv}^{\mathcal{E}}(\cdot,\cdot)$ (and thus does not use s^* as the first input to the inversion oracle $E^{-1}(\cdot,\cdot)$), then $\mathcal{S}^{\Lambda}$ cannot obtain any advantage from $\mathcal{A}^{\Lambda,\Omega}$ to invert $y^* = E(s^*, x^*)$. This is because for any $s \neq s^*$, $E(s^*, \cdot)$ is a random permutation which is independent from $E(s, \cdot)$. Since $\mathcal{A}^{\Lambda,\Omega}$ will only make a single query using $pk = s$ as the first input to $\Omega.\mathsf{Inv}^{\mathcal{E}}(\cdot,\cdot)$ after $\mathcal{S}_2^{\Lambda}$ correctly answers a random decryption query, it suffices to show that $\mathcal{S}_2^{\Lambda}$ cannot correctly answer the decryption query from $\mathcal{A}^{\Lambda,\Omega}$ with non-negligible probability when $s = s^*$.

Recall that given a security parameter κ and a public key $pk = s^*$ as inputs, $\mathcal{A}^{\Lambda,\Omega}$ works as follows: 1) set $V = \kappa\|s^*$ and compute $r = \Omega.\mathcal{R}_2(V)$; 2) send r as a G-oracle query to $\mathcal{S}_3^{\Lambda}(st_1)$ and obtain a response z_1; 3) update $V := V\|z_1$, and compute $\mu = \Omega.\mathcal{R}_1(V) \in \{0,1\}^{n_1}$; 4) send $x_0 = (\mu\|0_{n_2}) \oplus z_1$ as an H-oracle query to $\mathcal{S}_3^{\Lambda}(st_1)$ and obtain a response z_2; 5) send $y = \Lambda.\mathsf{Eval}^{\mathcal{E}}(s, x_0\|x_1))$ as a decryption query to $\mathcal{S}_2^{\Lambda}(pk, st_1)$ and obtain a result μ', where $x_1 = r \oplus z_2$. We now show that the probability that $\mu' = \mu$ is negligible. First, by the fact that $\Omega.\mathcal{R}_1$ and $\Omega.\mathcal{R}_2$ evaluates random functions, we have that r and μ are both uniformly random, and in particular is independent from the inputs (st_1, y) of $\mathcal{S}_2^{\Lambda}$. Second, since $\mathcal{S}_2^{\Lambda}$ is unaware of the RO queries made to G and H when answering the decryption query $y = \Lambda.\mathsf{Eval}^{\mathcal{E}}(s, x_0\|x_1)$, and

$$x_0\|x_1 = ((\mu\|0_{n_2}) \oplus z_1)\|(r \oplus z_2) = ((\mu\|0_{n_2}) \oplus G(r))\|(r \oplus H(x_0))$$

has min-entropy at least $2^{n_1} \geq 2^{\kappa}$ in the view of $\mathcal{S}_2^{\Lambda}$ (since μ is uniformly distributed over $\{0,1\}^{n_1}$), the probability that $\mathcal{S}_2^{\Lambda}$ can output $\mu' = \mu$ by making a polynomial number of queries to Λ is negligible (because $\mathcal{S}_2^{\Lambda}$ can only determine μ by making a query $(s^*, x_0\|x_1)$ to $\Lambda.\mathsf{Eval}^{\mathcal{E}}(\cdot,\cdot)$). In other words, $\mathcal{A}^{\Lambda,\Omega}$ will abort at step 6 with probability negligibly close to 1. Thus, $\mathcal{S}^{\Lambda}$ can only obtain negligible advantage from $\mathcal{A}^{\Lambda,\Omega}$ in inverting $y^* = E(s^*, x^*)$, which completes the proof of Theorem 11. □

Combining the results in Lemma 10 and Theorem 11, we have that NPReds does not imply CPReds, namely, NPReds $\not\Rightarrow$CPReds.

[10] Here, we allow the sub-algorithm $\mathcal{S}_3$ to access the oracle Λ, which does not conflict with the restriction that "$\mathcal{S}_3$ does not have access to $\mathrm{P}.\mathsf{Oral}(s_x, \cdot)$" in Definition 7, since Λ is an oracle which implement the trapdoor permutations and is assumed to be publicly known to all parties, while $\mathrm{P}.\mathsf{Oral}(s_x, \cdot)$ is assumed to be a private oracle which can only be accessed by the party who owns the secret value s_x. In fact, the problem we considered here is to invert the trapdoor permutations (i.e., the partial-domain one-wayness of Λ), which is a non-interactive problem (see Sec. 7 for the definition), i.e., the corresponding oracle "$\mathrm{P}.\mathsf{Oral}(s_x, \cdot)$" is actually a dummy one $\mathrm{P}.\mathsf{Oral}(s_x, \cdot) = \perp$.

7.2 CPRed $\not\Rightarrow$ RPRed

Let κ be the security parameter, and let ℓ_μ be positive integer. Let $\ell(\cdot)$ be an integer function. Let $\Pi_{\mathrm{TDP}} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv})$ be a trapdoor one-way permutation family with respect to $\ell(\cdot)$. Let $H : \{0,1\}^{\ell_\mu} \to \{0,1\}^{\ell(\kappa)}$ be a hash function. The signature scheme $\Pi_{\mathrm{FDH}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ given in [4] works as follows:

- $\mathsf{KeyGen}(1^\kappa)$: take a security parameter κ as input, compute and return the verification and signing key pair $(vk, sk) = (s, td) \leftarrow \Pi_{\mathrm{TDP}}.\mathsf{Gen}(1^\kappa)$;
- $\mathsf{Sign}(sk, \mu)$: take the signing key $sk = td$ and a message $\mu \in \{0,1\}^{\ell_\mu}$ as inputs, compute $y = H(\mu)$ and return the signature $\sigma = x = \Pi_{\mathrm{TDP}}.\mathsf{Inv}(td, y)$;
- $\mathsf{Verify}(vk, \mu, \sigma)$: take the verification key $vk = s$, a message $\mu \in \{0,1\}^{\ell_\mu}$ and a signature $\sigma = x \in \{0,1\}^{\ell(\kappa)}$ as inputs, first compute $y = H(\mu) \in \{0,1\}^{\ell(\kappa)}$. Then, returns 1 if $y = \Pi_{\mathrm{TDP}}.\mathsf{Eval}(s, x)$, else return 0.

As we have shown in Section 5.2, the FDH signature from TDP is provable under CPReds. Formally, we have the following theorem.

Theorem 12. *If the trapdoor permutation family Π_{TDP} is one-way, the signature scheme Π_{FDH} is provably EUF-CMA secure under CPReds.*

Besides, we also have the following lemma which is implicit in [14].

Lemma 11. *There is no RPRed from the EUF-CMA security of the signature scheme Π_{FDH} to the one-wayness of the trapdoor permutation family Π_{TDP}.*

This lemma directly follows from two facts in [14]: 1) a RPRed implies a reduction in the weakly programming ROM (WPROM); and 2) Π_{FDH} is not provable even against key-only attacks in the WPROM. We omit the details.

Combining the results in Theorem 12 and Lemma 11, we have that CPReds does not imply RPReds, namely, CPReds $\not\Rightarrow$RPReds.

References

1. Ananth, P., Bhaskar, R.: Non observability in the random oracle model. In: Susilo, W., Reyhanitabar, R. (eds.) Provable Security 2013. pp. 86–103. Springer, Heidelberg (2013)
2. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: CCS 1993. pp. 62–73. ACM (1993)
3. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: De Santis, A. (ed.) Advances in Cryptology – EUROCRYPT'94. pp. 92–111. Springer, Heidelberg (1995)
4. Bellare, M., Rogaway, P.: The exact security of digital signatures-how to sign with RSA and Rabin. In: Maurer, U. (ed.) Advances in Cryptology — EUROCRYPT '96. pp. 399–416. Springer, Heidelberg (1996)
5. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. Theoretical Computer Science 560, 7 – 11 (2014)
6. Biham, E., Boyer, M., Boykin, P.O., Mor, T., Roychowdhury, V.: A proof of the security of quantum key distribution. Journal of Cryptology 19(4), 381–439 (Oct 2006)
7. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D., Wang, X. (eds.) ASIACRYPT 2011, pp. 41–69. Springer, Heidelberg (2011)
8. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001, pp. 213–229. Springer (2001)
9. Boneh, D., Zhandry, M.: Secure signatures and chosen ciphertext security in a quantum computing world. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. pp. 361–379. Springer, Heidelberg (2013)
10. Brakerski, Z., Christiano, P., Mahadev, U., Vazirani, U., Vidick, T.: A cryptographic test of quantumness and certifiable randomness from a single quantum device. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS). pp. 320–331 (2018)
11. Brassard, G., HØyer, P., Tapp, A.: Quantum cryptanalysis of hash and claw-free functions. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN'98: Theoretical Informatics. pp. 163–169. Springer, Heidelberg (1998)

12. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. J. ACM 51(4), 557–594 (Jul 2004)
13. Dagdelen, Ö., Fischlin, M., Gagliardoni, T.: The Fiat–Shamir transformation in a quantum world. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology - ASIACRYPT 2013. pp. 62–81. Springer, Heidelberg (2013)
14. Fischlin, M., Lehmann, A., Ristenpart, T., Shrimpton, T., Stam, M., Tessaro, S.: Random oracles with(out) programmability. In: Abe, M. (ed.) Advances in Cryptology – ASIACRYPT 2010. pp. 303–320. Springer, Heidelberg (2010)
15. Fuchs, C.A., van de Graaf, J.: Cryptographic distinguishability measures for quantum-mechanical states. IEEE Transactions on Information Theory 45(4), 1216–1227 (May 1999)
16. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology 26(1), 80–101 (Jan 2013)
17. Fujisaki, E., Okamoto, T., Pointcheval, D., Stern, J.: RSA-OAEP is secure under the RSA assumption. Journal of Cryptology 17(2), 81–104 (Mar 2004)
18. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: STOC 2008. pp. 197–206. ACM (2008)
19. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: STOC 1996. pp. 212–219. ACM (1996)
20. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) Theory of Cryptography. pp. 341–371. Springer International Publishing, Cham (2017)
21. Hsiao, C.Y., Reyzin, L.: Finding collisions on a public road, or do secure hash functions need secret coins? In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. pp. 92–105. Springer, Heidelberg (2004)
22. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisted. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. pp. 96–125. Springer,Cham (2018)
23. Kiltz, E., Lyubashevsky, V., Schaffner, C.: A concrete treatment of Fiat-Shamir signatures in the quantum random-oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 552–586. Springer International Publishing, Cham (2018)
24. Mahadev, U.: Classical verification of quantum computations. In: 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS). pp. 259–267 (2018)
25. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) Theory of Cryptography. pp. 21–39. Springer, Heidelberg (2004)
26. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. pp. 111–126. Springer, Heidelberg (2002)
27. Nielsen, M.A., Chuang, I.: Quantum computation and quantum information (2000)
28. Saito, T., Xagawa, K., Yamakawa, T.: Tightly-secure key-encapsulation mechanism in the quantum random oracle model. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018. pp. 520–551. Springer, Cham (2018)
29. Shor, P.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (1997)
30. Shoup, V.: A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112 (2001)
31. Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In: Hirt, M., Smith, A. (eds.) Theory of Cryptography, pp. 192–216. Springer, Heidelberg (2016)
32. Unruh, D.: Quantum position verification in the random oracle model. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014. pp. 1–18. Springer, Heidelberg (2014)
33. Unruh, D.: Non-interactive zero-knowledge proofs in the quantum random oracle model. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015. pp. 755–784. Springer, Heidelberg (2015)
34. Unruh, D.: Revocable quantum timed-release encryption. J. ACM 62(6), 49:1–49:76 (Dec 2015)
35. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012, pp. 758–775. Springer, Heidelberg (2012)

A Randomly-Programming Reduction

The notion of randomly-programming reduction (RPRed) is formalized via the notion of randomly programmable random oracle (RPRO). Formally, a RPRO $\mathcal{O} = (R_{eval}, R_{rand}, R_{prog})$ is an idealized object consisting of a public interface R_{eval}, and two private interfaces R_{rand} and R_{prog}, where R_{eval} behaves as a conventional RO mapping $Dom \to Rng$; R_{rand} maps a string in $\{0,1\}^*$ to a random value in Rng; and $R_{prog}(X, Y)$ takes an $X \in Dom$ and a string $Y \in \{0,1\}^*$ as inputs, sets $R_{eval}(X) = R_{rand}(Y)$. As NPReds, a RPRed (i.e., a black-box reduction in the model equipped with a RPRO) is allowed to learn all the RO-queries made by the adversary $\mathcal{A}$ via the public interface R_{eval}, but unlike NPReds, a RPRed is also allowed to program the answer of a given RO-query X made by the adversary $\mathcal{A}$ via the two private interfaces R_{rand} and R_{prog}. Specifically, after receiving a RO query X from $\mathcal{A}$, the reduction can make a number of queries to R_{rand} and R_{prog} before returning the answer $R_{eval}(X)$ (which can be programmed by using the interface R_{prog}) to $\mathcal{A}$.

B Information Theory

In this section, we recall some definitions related to the Shannon entropy of random variables. Formally, let X, Y be two random variables with support D_X, D_Y, respectively. The entropy of X is defined as

$$H(X) = -\sum_{x \in D_X} \Pr[X = x] \log_2(\Pr[X = x]).$$

The entropy of X conditioned on $Y = y$ is defined as

$$H(X|y) = -\sum_{x \in D_X} \Pr[X = x|y] \log_2(\Pr[X = x|y]).$$

The entropy of X conditioned on random variable Y is defined as

$$H(X|Y) = \sum_{y \in D_Y} \Pr[Y = y] H(X|y).$$

The mutual information between X and Y is defined as

$$I(X, Y) = H(X) - H(X|Y).$$

Intuitively, the mutual information indicates the decrease in the entropy of X due to learning of Y, which is symmetric to X and Y.

C Definition and Security of Signatures

A digital signature scheme $\mathcal{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ consists of three PPT algorithms. Taking the security parameter κ as input, the key generation algorithm outputs a verification key vk and a secret signing key sk, i.e., $(vk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$. The signing algorithm takes vk, sk and a message $M \in \{0,1\}^*$ as inputs, outputs a signature σ on M, briefly denoted as $\sigma \leftarrow \mathsf{Sign}(sk, M)$. The verification algorithm takes vk, message $M \in \{0,1\}^*$ and a string $\sigma \in \{0,1\}^*$ as inputs, outputs 1 if σ is a valid signature on M, else outputs 0, denoted as $1/0 \leftarrow \mathsf{Verify}(vk, M, \sigma)$. For correctness, we require that for any $(vk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$, any message $M \in \{0,1\}^*$, and any $\sigma \leftarrow \mathsf{Sign}(sk, M)$, the equation $\mathsf{Verify}(vk, M, \sigma) = 1$ holds with overwhelming probability, where the probability is taken over the choices of the random coins used in KeyGen, Sign and Verify.

The standard security notion for digital signature scheme is the existential unforgeability against chosen message attacks (EUF-CMA), which (informally) says that any PPT forger, after seeing valid signatures on a polynomial number of adaptively chosen messages, cannot create a valid signature on a new message. Formally, consider the following game between a challenger $\mathcal{C}$ and a forger $\mathcal{F}$:

KeyGen. The challenger $\mathcal{C}$ first runs $(vk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$ with the security parameter κ. Then, it gives the verification key vk to the forger $\mathcal{F}$, and keeps the signing secret key sk to itself.

Signing. The forger $\mathcal{F}$ is allowed to ask the signature on any fresh message M. The challenger $\mathcal{C}$ computes and sends $\sigma \leftarrow \mathsf{Sign}(sk, M)$ to the forger $\mathcal{F}$. The forger can repeat this any polynomial times.

Forge. $\mathcal{F}$ outputs a message-signature pair (M^*, σ^*). Let Q be the set of all messages queried by $\mathcal{F}$ in the signing phase. If $M^* \notin Q$ and $\mathsf{Verify}(vk, M^*, \sigma^*) = 1$, the challenger $\mathcal{C}$ outputs 1, else outputs 0.

We say that $\mathcal{F}$ wins the game if the challenger $\mathcal{C}$ outputs 1. The advantage of $\mathcal{F}$ in the above security game is defined as $\mathrm{Adv}^{\text{euf-cma}}_{\mathcal{SIG},\mathcal{F}}(1^\kappa) = \Pr[\mathcal{C} \text{ outputs } 1]$.

Definition 9 (EUF-CMA Security). *Let κ be the security parameter. A signature scheme $\mathcal{SIG}$ is said to be existentially unforgeable against chosen message attacks (*EUF-CMA*) if the advantage* $\mathrm{Adv}^{\text{euf-cma}}_{\mathcal{SIG},\mathcal{F}}(1^\kappa)$ *is negligible in κ for any PPT forger $\mathcal{F}$.*

D Definition and Security of Identity-Based Encryption

An identity-based encryption (IBE) scheme consists of four PPT algorithms $\Pi_{\text{ibe}} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$. Taking the security parameter κ as input, the randomized key generation algorithm Setup outputs a master public key mpk and a master secret key msk, denoted as $(mpk, msk) \leftarrow \mathsf{Setup}(1^\kappa)$. The (randomized) extract algorithm takes mpk, msk and an identity id as inputs, outputs a user private key sk_{id} for id, briefly denoted as $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$. The randomized encryption algorithm Enc takes mpk, id and a plaintext M as inputs, outputs a ciphertext C, denoted as $C \leftarrow \mathsf{Enc}(mpk, id, M)$. The deterministic algorithm Dec takes sk_{id} and C as inputs, outputs a plaintext M, or a special symbol $\perp$, which is denoted as $M/\perp \leftarrow \mathsf{Dec}(sk_{id}, C)$. In addition, for all $(mpk, msk) \leftarrow \mathsf{Setup}(1^\kappa), sk_{id} \leftarrow \mathsf{Extract}(msk, id)$ and any plaintext M, we require that $\mathsf{Dec}(sk_{id}, C) = M$ holds for any $C \leftarrow \mathsf{Enc}(mpk, id, M)$.

The standard semantic security of IBE was first introduced in [8]. Formally, consider the following game played by an adversary $\mathcal{A}$.

Setup. The challenger $\mathcal{C}$ first runs $\mathsf{Setup}(1^\kappa)$ with the security parameter κ. Then, it gives the adversary $\mathcal{A}$ the master public key mpk, and keeps the master secret key msk to itself.

Phase 1. The adversary is allowed to query the user private key for any identity id. The challenger $\mathcal{C}$ runs $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$ and sends sk_{id} to the adversary $\mathcal{A}$. The adversary can repeat the user private key query any polynomial times for different identities.

Challenge. The adversary $\mathcal{A}$ outputs challenge plaintexts M_0^*, M_1^* and a challenge identity id^* with a restriction that id^* is not used in the user private key query in phase 1. The challenger $\mathcal{C}$ chooses a bit $b \in \{0, 1\}$. Then, it computes $C^* \leftarrow \mathsf{Enc}(mpk, id^*, M_b^*)$. Finally, it sends C^* to $\mathcal{A}$.

Phase 2. The adversary can adaptively make more user private key queries with any identity $id \neq id^*$. The challenger $\mathcal{C}$ responds as in Phase 1.

Guess. Finally, $\mathcal{A}$ outputs a guess $b \in \{0, 1\}$. If $b = b^*$, the challenger $\mathcal{C}$ outputs 1, else outputs 0.

The advantage of $\mathcal{A}$ in the above security game is defined as $\mathrm{Adv}^{\text{ind-id-cpa}}_{\Pi_{\text{ibe}},\mathcal{A}}(\kappa) = |\Pr[b = b^*] - \frac{1}{2}|$.

Definition 10 (IND-ID-CPA Security). *We say an IBE scheme Π_{ibe} is* IND-ID-CPA *secure if for any PPT adversary $\mathcal{A}$, its advantage* $\mathrm{Adv}^{\text{ind-id-cpa}}_{\Pi_{\text{ibe}},\mathcal{A}}(\kappa)$ *is negligible in κ.*

E Definition and Security of Public-Key Encryption

A PKE scheme Π_{PKE} consists of three algorithms $\Pi_{\text{PKE}} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$. $\mathtt{Gen}$ is a randomized algorithm which takes a security parameter k as input, outputs a key pair (pk, sk). The probabilistic algorithm $\mathtt{Enc}$ takes as input a public key pk and a message m, returns a ciphertext c of m. $\mathtt{Dec}$ is a deterministic algorithm that takes as input a ciphertext c and a secret sk, outputs a message m or a special symbol $\perp$. We now recall the IND-CCA security of PKE scheme. Consider the following game between a challenger and an adversary $\mathcal{A}$.

Setup: Given the security parameter k, the challenger generates $(pk, sk) \leftarrow \texttt{Gen}(1^k)$ and gives the public key pk to $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ may adaptively make a number of decryption queries on ciphertext c, the challenger responds with $\texttt{Dec}(sk, c)$.

Challenge: At some point, $\mathcal{A}$ outputs two equal-length messages m_0, m_1. The challenger chooses a random bit $b \in \{0, 1\}$ and returns $c^* \leftarrow \texttt{Enc}(pk, m_b)$.

Phase 2: The adversary $\mathcal{A}$ makes more decryption queries as in phase 1, but with a constraint that decryption queries on c^* are not allowed.

Guess: Eventually, The adversary $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$.

The adversary wins the game if $b = b'$. The advantage of $\mathcal{A}$ in the above game is defined as $\text{Adv}^{\text{ind-cca}}_{\Pi_{\text{PKE}},\mathcal{A}}(k) = |Pr[b = b'] - 1/2|$.

Definition 11 (IND-CCA). *We say that a PKE scheme Π_{PKE} is secure under chosen-ciphertext attacks (IND-CCA), if for any polynomial time adversary $\mathcal{A}$, its advantage in the above game is negligible.*