How to Record Quantum Queries, and Applications to Quantum Indifferentiability

Abstract. The quantum random oracle model (QROM) has become the standard model in which to prove the post-quantum security of random-oracle-based constructions. Unfortunately, none of the known proof techniques allow the reduction to record information about the adversary's queries, a crucial feature of many classical ROM proofs, including all proofs of indifferentiability for hash function domain extension.

In this work, we give a new QROM proof technique that overcomes this "recording barrier". We do so by giving a new "compressed oracle" which allows for efficient on-the-fly simulation of random oracles, roughly analogous to the usual classical simulation. We then use this new technique to give the first proof of quantum indifferentiability for the Merkle-Damgård domain extender for hash functions. We also give a proof of security for the Fujisaki-Okamoto transformation; previous proofs required modifying the scheme to include an additional hash term. Given the threat posed by quantum computers and the push toward quantum-resistant cryptosystems, our work represents an important tool for efficient post-quantum cryptosystems.

1 Introduction

The random oracle model [BR93] has proven to be a powerful tool for heuristically proving the security of schemes that otherwise lacked a security proof. In the random oracle model (ROM), a hash function H is modeled as a truly random function that can only be evaluated by querying an oracle for H. A scheme is secure in the ROM if it can be proven secure in this setting. Of course, random oracles cannot be efficiently realized; in practice, the random oracle is replaced with a concrete efficient hash function. The hope is that the ROM proof will indicate security in the real world, provided there are no structural weaknesses in the concrete hash function.

Meanwhile, given the looming threat of quantum computers [IBM17], there has been considerable interest in analyzing schemes for so called "post-quantum" security [NIS17, Son14, ATTU16, CBH⁺17, YAJ⁺17, CDG⁺17, CDG⁺15]. Many of the proposed schemes are random oracle schemes; Boneh et al. [BDF⁺11] argue that the right way of modeling the random oracle in the quantum setting is to use the quantum random oracle model, or QROM. Such a model allows a quantum attacker to query the random oracle on a quantum superposition of inputs. The idea is that a real-world quantum attacker, who knows the code for the concrete hash function, can evaluate the hash function in superposition in order to perform tasks such as Grover search [Gro96] or collision finding [BHT98]. In order to accurately capture such real-world attacks, it is crucial to model the random oracle to allow for such superposition queries. The quantum random oracle model has been used in a variety of subsequent works to prove the post-quantum security of cryptosystems [BDF$^+$11, Zha12b, Zha15, TU16, Eat17].

The Recording Barrier. Unfortunately, proving security in the quantum random oracle model can be extremely difficult. Indeed, in the classical random oracle model, one can copy down the adversary's queries as a means to learning what points the adversary is interested in. Many classical security proofs crucially use this information in order to construct a new adversary which solves some hard underlying problem, reaching a contradiction. In the quantum setting, such copying is impossible by no-cloning. One can try to record some information about the query, but this amounts to a measurement of the adversary's query state which can be detected by the adversary. A mischievous adversary may refuse to continue if it detects such a measurement, rendering the adversary useless for solving the underlying problem. Because of the difficulty in reading an adversary's query, it also becomes hard to adaptively program the random oracle, another common classical proof technique.

This difficulty has led authors to develop new quantum-sound proof techniques to replace classical techniques, such as Zhandry's small-range distributions [Zha12a] or Targhi and Unruh's extraction technique [TU16]. These proof techniques choose the oracle from a careful distribution that allows for proofs to go through. However, every such proof technique always chooses a classical oracle at the beginning of the experiment, and leave the oracle essentially unchanged through the entire execution. The inability to change the oracle seems inherent, since if the proof gives the adversary different oracles during different queries, this is potentially easily detectable (even by classical adversaries)[1]

Constraining the oracles to be fixed functions seems to limit what can be proved using such non-recording techniques. For example, Dagdelen, Fischlin, and Gagliardoni [DFG13] show that such natural proof techniques are likely incapable of proving the security of Fiat-Shamir[2]. This leads to a natural question:

Is it possible to record information about an
adversary's quantum query without the adversary detecting

Enter Indifferentiability. The random oracle model (quantum or otherwise) assumes the adversary treats the hash function as a monolithic object. Unfortunately, hash functions in practice are usually built from smaller building blocks, called compression functions. If one is not careful, hash functions built in this way are vulnerable to attacks such as length-extension attacks. Coron et al. [CDMP05]

[1] The one exception we are aware of is Unruh's adaptive programming [Unr15]. This proof does change the oracle adaptively, but only inputs for which adversary's queries have only negligible "weight". Thus, the change is not detectable. The following discussion also applies to Unruh's technique.

[2] We note that if the underlying building blocks are strengthened, Fiat-Shamir was proven secure by Unruh [Unr16]

show that a hash function built from a compression function can be as good as a monolithic oracle in many settings if it satisfies a notion of *indifferentiability*, due to Maurer, Renner, and Holenstein [MRH04]. Roughly, in indifferentiability, an adversary A has oracle access to both h and H, and the adversary is trying to distinguish two possible worlds. In the "real world", h is a random function, and H is built from h according to the hash function construction. In the "ideal world", H is a random function, and h is simulated so as to be consistent with H. A hash function is indifferentiable from a random oracle if no efficient adversary can distinguish the two worlds.

Coron et al.'s proof of indifferentiability for Merkle-Damgard requires the simulator to remember the queries that the adversary has made. This is actually inherent for any domain extender, by a simple counting argument discussed below. In the quantum setting, such recording presents a serious issue, as recording a query is equivalent (from the adversary's point of view) to measuring the query. As any measurement will disturb the quantum system, such measurement may be detectable to the adversary. Note that in the case where A is interacting with a truly random h, there is no measurement happening. Therefore, if such a measurement can be detected, the adversary can distinguish the two cases, breaking indifferentiability.

Example. To illustrate what might go wrong, we will use the simple example from Coron et al. [CDMP05]. Here, we will actually assume access to two independent compression functions $h_0, h_1 : \{0,1\}^{2n} \to \{0,1\}^n$. We will define $H : \{0,1\}^{3n} \to \{0,1\}^n$ as $H(x, y) = h_1(h_0(x), y)$, where $x \in \{0,1\}^{2n}, y \in \{0,1\}^n$.

To argue that H is indifferentiable from a random oracle, Coron et al. use the following simulator S, which has access to H, and tries to implement the oracles h_0, h_1. S works as follows:

- S keeps databases D_0, D_1, which will contain tuples (x, y). D_b containing (x, y) means that S has set the $h_b(x) = y$.
- h_0 is implemented on the fly: every query on x looks up $(x, y) \in D_0$, and returns y if it is found; if no such pair is found, a random y is chosen and returned, and (x, y) is added to D_0.
- By default, h_1 is answered randomly on the fly as in h_0. However, it needs to make sure that $h_1(h_0(x), y)$ always evaluates to $H(x, y)$, else it is trivial to distinguish the two worlds. Therefore, on a query (z, y), h_1 will check if there is a pair (x, z) in D_0 for some x. If so, it will reasonably guess that the adversary is trying to evaluate $H(x, y)$, and respond by making a query to $H(x, y)$. Otherwise it will resort to the default simulation.

Note that by defining the simulator in this way, if the adversary ever tries to evaluate H on (x, z) by first making a query x to h_0 to get y, and then making a query (y, z) to h_1, the simulator will correctly set the output of h_1 to $H(x, z)$, so that the adversary will get a result that is consistent with H. However, note that it is crucial that S wrote down the queries made to h_0, or else it will not know which point to query H when simulating h_1.

Now consider a quantum adversary. A quantum query to, say, h_0 will be the following operation:

$$\sum_{x\in\{0,1\}^{2n},u\in\{0,1\}^n} \alpha_{x,u}|x,u\rangle \mapsto \sum_{x\in\{0,1\}^{2n},u\in\{0,1\}^n} \alpha_{x,u}|x,u\oplus h_0(x)\rangle$$

Now, imagine our simulator trying to answer queries to h_0 in superposition. For simplicity, suppose this is the first query to h_0, so D_0 is empty. The natural approach is to just have S store its database D_0 in superposition, performing a map that may look like $|x,u\rangle \mapsto |x,u\oplus y\rangle \otimes |x,y\rangle$, where y is chosen randomly, and everything to the right of the $\otimes$ is the simulators state.

But now consider the following query by an adversary. It sets up the uniform superposition $\sum_{x,u}|x,u\rangle$ and queries. In the case where h_0 is a classical function, then this state becomes

$$\sum_{x,u}|x,u\oplus h_0(x)\rangle = \sum_{x,u}|x,u\rangle$$

Namely, the state is unaffected by making the query. In contrast, the simulated query would result in

$$\sum_{x,u}|x,u\oplus y\rangle \otimes |x,y\rangle$$

Here, the adversary's state is now entangled with the simulator's. It is straightforward to detect this entanglement by applying the Quantum Fourier Transform (QFT) to the adversary's x registers, and then measuring the result. In the case where the adversary is interacting with a random h_0, the QFT will result in a 0. In the simulated case, the QFT will result in a random string. These two cases are therefore easily distinguishable.

To remedy this issue, prior works in the quantum regime have abandoned on-the-fly simulation, instead opting for stateless simulation. Here, the simulator commits to a function to implement the oracle in the very beginning, and then sticks with this implementation throughout the entire experiment. Moreover, the simulator never records any information about the adversary's query, lest the adversary detect the entanglement with the simulator. This will certainly fix the issue above, and by carefully choosing the right implementations prior works have shown how to translate many classical results into the quantum setting.

However, for indifferentiability, choosing a single fixed function for h_0 introduces new problems. Now when the adversary makes a query to h_1, the simulator needs to decide if the query represents an attempt at evaluating H, and if so, it must program the output of h_1 accordingly. However, without knowing what inputs the adversary has queried to h_0, it seems impossible for the simulator to determine which point the adversary is interested in. For example, if the adversary queries h_1 on (y,z), there will be roughly 2^n possible x that gave rise to this y (since h_0 is compressing). Therefore, the simulator must choose from one of 2^n inputs of the form (x,z) on which to query H.

To make matters even more complicated, an adversary can submit the uniform superposition $\sum_x |x, 0\rangle$, resulting in the state $\sum_x |x, h_0(x)\rangle$, which causes it to "learn" $y = h_0(x)$. At this point, the simulator should be ready to respond to an h_1 query on (y, z) by using x, meaning the simulator *must* be entangled with x. Then, at some later time, the adversary can query again on the state $\sum_x |x, h_0(x)\rangle$, resulting in the original state $\sum_x |x, 0\rangle$ again. The adversary can test that it received the correct state using the quantum Fourier transform. Therefore, after this later query, the simulator must be un-entangled with x. Even more complex strategies are possible, where the adversary can compute and un-compute h_0 in stages, so as to try to hide what it is doing from any potential simulator.

These issues are much more general than just the simple domain extender above. Indeed, even classically domain extension with a *stateless* simulator is *impossible*, by the following simple argument. Suppose there is a hash function $H : \{0,1\}^M \to \{0,1\}^N$ built from a compression function $h : \{0,1\}^m \to \{0,1\}^n$. Let $L = M + \log_2 N, \ell = m + \log_2 n$. Then L, ℓ represent the logarithm of the size of the truth tables for H, h. Since we are domain extending, we are interested in the case where $L \gg \ell$. Suppose even $L \geq \ell + 0.001$.

Suppose toward contradiction that h can be simulated statelessly, which we will represent as Sim^H (since the function can make H queries). Then h has a truth table of size 2^ℓ. In the real world, H agrees with C^h on all inputs; therefore in order for indifferentiability to hold, in the simulated world a uniformly random H must agree with $C^h = C^{\mathsf{Sim}^H}$ on an overwhelming fraction of inputs. But this is clearly impossible, as it would allow us to compress the random truth table of H: simply output the truth table for Sim^H, along with the ϵ fraction of of input/output pairs where H and C^{Sim^H} disagree. The total length of this compressed truth table is $2^\ell + (\epsilon 2^M)(MN) = 2^\ell + \epsilon N 2^L$. As ϵ is negligible (and therefore much smaller than $1/N$) the compressed truth table will be smaller than 2^L, the size of the truth table for H. But since H is a random function its truth table cannot be compressed, reaching a contradiction.

Therefore, any simulator for indifferentiability, regardless of the scheme, *must* inherently store information about the adversary. But the existing QROM techniques are utterly incapable of such recording. We therefore ask:

Is indifferentiable domain extension even possible?

1.1 This Work

In this work, perhaps surprisingly, we answer the question above in the affirmative. Namely, we give a new *compressed oracle technique*, which allows for recording the adversary's queries in a way that the adversary can never detect. The intuition is surprisingly simple: an adversary interacting with a random oracle can be thought of as being entangled with a uniform superposition of oracles. As entanglement is symmetric, if the adversary ever has any information about the oracle, the *oracle must also have information about the adversary.* Therefore a simulator can always record *some* information about the adversary, if done carefully.

We then use the technique to prove the indifferentiability of the Merkle-Damgård construction. We believe our new technique will be of independent interest; for example our technique can be used to prove the security of the Fujisaki-Okamoto transformation [FO99], and also gives very short proofs of several quantum query lower bounds.

The Compressed Oracle Technique. In order to prove indifferentiability, we devise a new way of analyzing quantum query algorithms

Consider an adversary interacting with an oracle $h : \{0,1\}^m \to \{0,1\}^n$. It is well established that the usual quantum oracle mapping $|x,y\rangle \mapsto |x, y \oplus h(x)\rangle$ is equivalent to the "phase" oracle, which maps $|x,u\rangle \mapsto (-1)^{u \cdot h(x)}|x,u\rangle$ (we discuss this equivalence in Section 3). For simplicity, in this introduction we will focus on the phase oracle, which is without loss of generality.

Next, we note that the oracle h being chosen at random is equivalent (from the adversary's point of view) to h being in uniform superposition $\sum_h |h\rangle$. Indeed, the superposition can be reduced to a random h by measuring, and measuring the h registers (which is outside of A's view) is undetectable to A. To put another way, the superposition over h is a *purification* of the adversary's mixed state.

Therefore, we will imagine the h oracle as containing $\sum_h |h\rangle$. When A makes a query on $\sum_{x,u} \alpha_{x,u}|x,u\rangle$, the joint system of the adversary and oracle are

$$\sum_{x,u} \alpha_{x,u}|x,u\rangle \otimes \sum_h |h\rangle$$

The query introduces a phase term $(-1)^{u \cdot h(x)}$, so the joint system becomes

$$\sum_{x,u} \alpha_{x,u}|x,u\rangle \otimes \sum_h |h\rangle(-1)^{u \cdot h(x)}$$

We normally think of the phase as being returned to the adversary, but the phase really affects the entire system, so it is equivalent to think of the phase as being added to the oracle's state.

Now, we will think of h as a vector of length $2^m \times n$ by simply writing down h's truth table. We will think of each x, u pair as a point function $P_{x,u}$ which outputs u on x and 0 elsewhere. Using our encoding of functions as vectors, we can write $u \cdot h(x)$ as $P_{x,u} \cdot h$. We can therefore write the post-query state as

$$\sum_{x,u} \alpha_{x,u}|x,u\rangle \otimes \sum_h |h\rangle(-1)^{h \cdot P_{x,u}}$$

In general, the state after making q queries can be written as

$$\sum_{x_1,\dots,x_q,u_1,\dots,u_q} \alpha_{x_1,\dots,x_q,u_1,\dots,u_q}|\psi_{x_1,\dots,x_q,u_1,\dots,u_q}\rangle \otimes \sum_h |h\rangle(-1)^{h \cdot (P_{x_1,u_1} + \cdots + P_{x_q,u_q})}$$

Next, notice that by applying the Quantum Fourier transform to h, the h registers will now contain $(P_{x_1,u_1} + \cdots + P_{x_q,u_q}) \bmod 2$. Working in the Fourier domain, we see that each query simply adds $P_{x,u}$ (modulo 2) to the result. In the Fourier domain, the initial state is 0.

Therefore, from A's point of view, it is indistinguishable whether the oracle for h is a random oracle, or it is implemented as follows:

- The oracle keeps as state a vector $D \in \{0,1\}^{n \times 2^m}$, initially set to 0.
- On any oracle query, the oracle performs the map $|x,u\rangle \otimes |D\rangle \mapsto |x,u\rangle \otimes |D \oplus P(x,u)\rangle$

Thus, with this remarkably simple change in perspective, the oracle can actually be implemented by recording and updating phase information about the queries being in made.

We can now take this a couple steps further. Notice that after q queries, D is non-zero on at most q inputs (since it is the sum of q point functions). Therefore, we can store the database in an extremely compact form, namely the list of (x,y) pairs where $y = D(x)$ and $y \neq 0$. Notice that this allows us to efficiently simulate a random oracle, without an a priori bound on the number of queries. Previously, simulating an unbounded number of queries efficiently required computational assumptions, and simulation was only computationally secure. In contrast, simulating random oracles exactly required $2q$-wise independent functions [Zha12b] and hence required knowing q up front. We therefore believe this simulation will have independent applications for the efficient simulation of quantum oracles. We will call this the compressed Fourier oracle.

We can then take our compressed Fourier oracle, and convert it back into a primal-domain oracle. Namely, for each (x,y) pair, we perform the QFT on the y registers. The result is a superposition of databases of (x,w) pairs, where w approximately represents $h(x)$. For any pair not in the database, $h(x)$ is implicitly a uniform superposition of inputs independent of the adversary's view. We call this the compressed standard oracle. It intuitively represents what the adversary knows about the function h: if (x,y) is in the database then the adversary "knows" $h(x) = y$, and otherwise, the adversary "knows" nothing about $h(x)$. In Section 3, we show how to directly obtain the compressed standard oracle.

Applying Compressed Oracles to Indifferentiability. The compressed standard oracle offers a simple way to keep track of the queries the adversary has made. In particular, it tracks exactly the kind of information needed in the classical indifferentiability proof above, namely whether or not a particular value has been queried by the adversary, and what the value of the oracle at that point is. We use this to give a quantum indifferentiability proof for Merkle-Damgård construction using prefix-free encodings [CDMP05].

To illustrate our ideas, consider our simple example above with h_0, h_1 and H. Our simulator will simulate h_0 as in the compressed standard oracle, keeping a (superposition over) list D_0 of (x,y) pairs. Next, our simulator must handle h_1 queries. When given a phase query $|y,z\rangle$, the simulator does the following. If first looks for a pair (x,y') in D_0 with $y' = y$. If one is found, it reasonably guesses that the adversary is interested in computing $H(x,z)$, and so it makes a query on (x, z) to H. Otherwise, it is reasonable to guess that the adversary is not trying to compute H on any input, since the adversary does not "know" any inputs to h_0 that would result in a query to h_1 on (y, z).

While the above appears to work, we need to make sure the simulator does not disturb the compressed oracle. Unfortunately, some disturbance is necessary. Indeed, determining the value of $h_0(x)$ is a measurement in the primal domain. On the other hand, the update procedure for the compressed oracles needs to decide whether or not x belongs in the database, and this corresponds to a measurement in the *Fourier* domain (since in the Fourier domain, $h_0(x)$ must be non-zero). These two measurements do not commute, so by the uncertainty principle it is impossible to perform both measurements perfectly.

Nonetheless, we show that the errors are small. Intuitively, we observe that the simulator does not actually need to know the entire value of $h_0(x)$, just whether or not it is equal to y. We call such information a "test". Similarly, the compressed oracle implementation just needs to know whether or not $h_0(x)$ is equal to 0, but in the Fourier domain.

Now, these primal and Fourier tests still do not commute. Fortunately, they "almost" commute, which we formalize in Appendix E. The intuition is that, if a primal test of the form "is $h_0(x) = y$" has a non-negligible chance of succeeding, $h_0(x)$ must be very "far" from the uniform superposition. This is because a uniform superposition puts an exponentially small weight on every outcome. Recall that the uniform superposition maps to $h_0(x) = 0$ in the Fourier domain. Thus by being "far" from uniform, the Fourier domain test has a negligibly-small chance of succeeding. Therefore, one of the two tests is always "almost" determined, meaning the measurement negligibly affects the state. This means that, no matter what initial state is, the two tests "almost" commute.

Thus, the simulator can perform these tests without perturbing the state significantly. This shows that h_0 queries are correctly simulated; we also need to show that h_1 queries are correctly simulated and consistent with H. The intuition above suggests that h_1 should be consistent with H, and indeed in Section 5 we show this using a careful sequence of hybrids. Then in Section D, we use the same ideas to prove the indifferentiability of Merkle-Damgård.

The Power of Forgetting. Surprisingly, our simulator ends up strongly resembling the classical simulator. It is natural to ask, therefore, how the simulator gets around the difficulties outlined above.

First, notice that if we translate the query $\sum_{x,u} |x, u\rangle$ in our example to a phase query, it becomes $\sum_x |x, 0\rangle$. This query has no effect on the oracle's state. This means the oracle remains un-entangled with the adversary, as desired.

Second, a query $\sum_x |x, 0\rangle$ becomes $\sum_{x,u} |x, u\rangle$ for a phase query. Consider applying the query to the compressed Fourier oracle. The joint quantum system of the adversary and simulator becomes

$$\sum_{x,u\neq 0} |x, u\rangle |\{(x, u)\}\rangle + \sum_x |x, 0\rangle |\{\}\rangle$$

A similar expression holds for the compressed standard oracle. Note that the simulator can clearly tell (whp) that the adversary has queried on x. Later, when the adversary queries on the same state a second time, (x, u) will get mapped to $(x, 0)$, and will hence be removed from the database. Thus, after this later query, the database contains no information about x. Hence, the adversary is un-entangled with x, and so it's tests will output the correct value.

Ultimately then, the key difference between our simulator and the natural quantum analog of the classical simulator is that our simulator must be ready to *forget* some of the oracle points it simulated previously. By implementing h_0 as a compressed oracle, it will forget *exactly* when it needs to so that the adversary can never detect that it is interacting with a simulated oracle.

Other results We expect our compressed oracle technique will have applications beyond indifferentiability. Here, we list two additional sets of results we are able to obtain using our technique:

Post-quantum security of Fujisaki-Okamoto. The Fujisaki-Okamoto transform [FO99] transforms a weak public key encryption scheme into a public key encryption scheme that is secure against *chosen ciphertext attacks*, in the random oracle model. Unfortunately, the classical proof does not work in quantum random oracle model, owing to similar issues with indifferentiability proofs. Namely, in one step of the proof, the reduction looks at the queries made by the adversary in order to decrypt chosen ciphertext queries. This is crucial to allow the reduction to simulate the view of the adversary without requiring the secret decryption key. But in the quantum setting, it is no longer straightforward to read the adversary's queries without disrupting its state.

Targhi and Unruh [TU16] previously modified the transformation by including an additional random oracle hash in the ciphertext. In the proof, the hash function is set to be injective, and the reduction can invert the hash in order to decrypt.

In Section F, we show how to adapt our compressed oracle technique to prove the security of the original transform without the extra hash. In addition, we show security against even *quantum* chosen ciphertext queries, thus proving security in the stronger model of Boneh and Zhandry [BZ13]. We note that recently, Jiang et al. [JZC$^+$18] proved the security of the FO transformation when used as a key encapsulation mechanism. Their proof is tight, whereas ours is somewhat loose. On the other hand, we note that their proof does not apply if FO is used directly as an encryption scheme, and does not apply in the case of quantum chosen ciphertext queries.

Simple Quantum Query Complexity Lower Bounds. We also show that our compressed oracles can be used to give very simple and optimal quantum query complexity lower bounds for problems for *random functions*, such as pre-image search, collision finding, and more generally k-SUM.

Our proof strategy is roughly as follows. First, since intuitively the adversary has no knowledge of values of h outside of D, except with very small probability any successful algorithm will output points in D. Therefore it suffices to bound the number of queries required to get D to contain a pre-image/collision/k-sum.

For pre-image search, we re-prove the optimal lower bound of $\Omega(2^{n/2})$ queries of [BBBV97], but for random functions; note that pre-image search for random functions and worst-case functions is equivalent using simple reductions. The proof appears superficially similar to [BBBV97]: we show that each query can increase the "amplitude" on "good" databases by a small $O(2^{-n/2})$ amount. After q queries, this amplitude becomes $O(q/2^{n/2})$, which we then square to get the probability of a "good" database. The proof is only slightly over a page once the compressed oracle formalism has been given.

We then re-prove the optimal collision lower bound of $\Omega(2^{n/3})$ queries for random functions, matching the worst case bound [AS04] and the more recent average case bound [Zha15]. Remarkably, our proof involves only a few lines of modification to the pre-image lower bound. We show that the amplitude on "good" databases increases by $O(\sqrt{q} \times 2^{n/2})$ for each query, where the extra $\sqrt{q}$ intuitively comes from the fact that the database has size at most q, giving q opportunities for a collision every time a new entry is added to the database[3].

In contrast to our very simple extension, the prior collision bounds involved very different techniques and were much more complicated. Also note that prior works could not prove directly that finding collisions were hard. Instead, they show that distinguishing a function with many collisions from an injective function was hard. This then only works directly for expanding functions, which are of little interest to cryptographers. Zhandry [Zha15] shows for random functions a reduction from expanding functions to compressing functions, giving the desired lower bound for compressing functions. Our proof, in contrast, works directly with functions of arbitrary domain and range. These features suggests that our proof technique is fundamentally different than those of prior works.

By generalizing our collision bound slightly, we can obtain an $\Omega(2^{n/(k+1)})$ lower bound for finding a set of distinct points $x_1, \ldots, x_k$ such that $\sum_i H(x_i) = 0$. This bound is tight as long as $n \leq km$ by adapting the collision-finding algorithm of [BHT98] to this problem. Again, our proof is obtained by modifying just a few lines of the pre-image search proof.

1.2 Related Works

Ristenpart, Shacham, and Shrimpton [RSS11] shows that indifferentiability is insufficient for replacing a concrete hash function with a random oracle in the setting of multi-stage games. Nonetheless, Mittelbach [Mit14] shows that indifferentiability can still be useful in these settings. Exploring the quantum analogs of these results is an interesting direction for future research.

[3] and the square root comes from the fact that the norm of the sum of q unit vectors of disjoint support is $\sqrt{q}$

2 Preliminaries

Distinguishing quantum states. The density matrix captures all statistical information about a mixed state. That is, if two states have the same density matrix, then they are perfectly indistinguishable.

For density matrices ρ, ρ' that are not identical, we define the trace distance as $T(\rho, \rho') = \frac{1}{2}\sum_i |\lambda_i|$, where λ_i are the eigenvalues of $\rho - \rho'$. The trace distance captures the maximum distinguishing advantage amongst all possible measurements of the state.

We will need the following Theorem of Bennett et al. (which we have slightly improved, see Appendix B.1 for the improved proof):

Lemma 1 ([BBBV97]). *Let $|\phi\rangle$ and $|\psi\rangle$ be quantum states with Euclidean distance ϵ. Then $T(|\phi\rangle\langle\phi|, |\psi\rangle\langle\psi|) = \epsilon\sqrt{1 - \epsilon^2/4} \leq \epsilon$.*

We will also need the following relaxation of commuting operations:

Definition 1. *Let U_0, U_1 be unitaries over the same quantum system. We say that U_0, U_1 ϵ-almost commute if, for any initial state ρ, the images of ρ under U_0U_1 and U_1U_0 are at most ϵ-far in trace distance.*

3 Oracle Variations

Here, we describe several oracle variations. The oracles will all be equivalent; the only difference is that the oracle registers and/or the query registers are encoded in different ways between queries. We start with the usual quantum random oracle, which comes in two flavors that we call the *standard oracle* and *phase oracle*. Then we will give our *compressed standard and phase oracles.*

Standard Oracle. Here, the oracle $H : \{0,1\}^m \to \{0,1\}^n$ is represented as its truth table: a vector of size 2^m where each component is an n-bit string.

The oracle takes as input a state consisting of three sets of registers: m-qubit x registers representing inputs to the function, n-qubit y registers for writing the response, and $n2^m$-qubit H registers containing the truth table of the actual function. The x, y registers come from the adversary, and the H registers are the oracle's state, which is hidden from the adversary accept by making queries. On basis states $|x, y\rangle \otimes |H\rangle$, the oracle performs the map $|x, y\rangle \otimes |H\rangle \mapsto |x, y \oplus H(x)\rangle \otimes |H\rangle$

For initialization, the oracle H will be initialized to the uniform superposition over all H: $\frac{1}{\sqrt{2^m \times 2^n}}\sum_H |H\rangle$. We will call this oracle StO.

The only difference between StO and the usual quantum random oracle model is that, in the usual model, H starts out as a uniformly chosen random function rather than a superposition (that is, the H registers are the completely mixed state). We will call the oracle with this different initialization StO'.

Lemma 2. StO *and* StO' *are perfectly indistinguishable. That is, for any adversary A making oracle queries, let $A^{\mathsf{StO}}()$ and $A^{\mathsf{StO}'}()$ denote the algorithm interfacing with* StO *and* StO'*, respectively. Then* $\Pr[A^{\mathsf{StO}}() = 1] = \Pr[A^{\mathsf{StO}'}() = 1]$

Proof. This can be seen by tracing out the oracle registers. The mixed state of the adversary in both cases will be identical. □

Thus, our initialization is equivalent to H being a uniformly random oracle.

Phase Oracle. We will also consider the well-known phase model of oracle queries. This model technically offers a different interface to the adversary, but can be mapped to the original oracle by simple Hadamard operations.

The oracle takes as input a state consisting of three sets of registers: x registers representing inputs to the function, z phase registers, and H registers containing the truth table of the actual function. On basis states $|x, y\rangle \otimes |H\rangle$, it performs the map $|x, z\rangle \otimes |H\rangle \mapsto (-1)^{y \cdot H(x)}|x, z\rangle \otimes |H\rangle$.

For initialization, H is the uniform superposition as before. We will call this oracle PhO. Analogous to the above, this is equivalent to the case where H is uniformly random. The following Lemma is implicit in much of the literature on quantum-accessible oracles:

Lemma 3. *For any adversary A making queries to StO, let B be the adversary that is identical to A, except it performs the Hadamard transformation $\mathsf{H}^{\otimes n}$ to the response registers before and after each query. Then* $\Pr[A^{\mathsf{StO}}() = 1] = \Pr[B^{\mathsf{PhO}}() = 1]$

Compressed Standard Oracles. We now define our compressed standard oracles. The intuition for our compressed standard oracle is the following. Let $|\tau\rangle$ be the uniform superposition. In the standard (uncompressed) oracle, suppose for each of the 2^m output registers, we perform the computation mapping $|\tau\rangle \mapsto |\tau\rangle|1\rangle$ and $|\phi\rangle \mapsto |\phi\rangle|0\rangle$ for any $|\phi\rangle$ orthogonal to $|\tau\rangle$. In other words, this computation tests whether or not the state of the output registers is 0 in the Fourier basis. We will write the output of the computation in some auxiliary space. Now the state of the oracle is a superposition over truth tables, and a superposition over vectors in $\{0,1\}^{2^m}$ containing the output of the tests. A straightforward exercise (and a consequence of our analysis below) shows that if we perform these tests after q queries, all vectors in the test vector superposition have at most q positions containing a 0. The reason is, roughly, if we do the tests before any queries the vector will be identically 1 since we had a uniform superposition (which is 0 in the Fourier basis). Then, each query affects only one position of the superposition, increasing the number of 0's by at most 1.

Also notice that anywhere the vector contains a 1, the corresponding truth table component contains exactly the uniform superposition $|\tau\rangle$. Anywhere the vector contains a 0, the corresponding truth table component contains a state that is guaranteed to be orthogonal to $|\tau\rangle$.

What we can do then is compress this overall state. We will simply write down all the positions where the test vector contained a 0, and keep track of the truth table component for that position. Everywhere else we can simply ignore since we know what the truth table contains. The result is a (superposition over) database consisting of at most q input/output pairs.

In more detail, a database D will be a collection of (x, y) pairs, where $(x, y) \in D$ means the function has been specified to have value y on input x. We will write $D(x) = y$ in this case. If, for an input x there is no pair $(x, y) \in D$, then we will write $D(x) = \perp$, indicating that the function has not been specified. We will maintain that a database D only contains at most one pair for a given x.

Concretely, if we have an upper bound t on the number of specified points, a database D will be represented an element of the set S^t, where $S = (\{0,1\}^m \cup \{\perp\}) \times \{0,1\}^n$. Each value in S is an (x, y) pair; if $x \neq \perp$ the pair means $D(x) = y$, and $x = \perp$ means the pair is unused. For $x_1 < x_2 < \cdots < x_\ell$ and $y_1, \ldots, y_\ell$, the database representing that input x_i has been set to y_i for $i \in [\ell]$, with all other points unspecified, will be represented as:

$$((x_1, y_1), (x_2, y_2), \ldots, (x_\ell, y_\ell), (\perp, 0^n), \ldots, (\perp, 0^n))$$

where the number of $(\perp, 0^n)$ pairs is equal to $t - \ell$.

After query q, the state of the oracle will be a superposition of databases in this form, using the upper bound $t = q$. So initially the state is empty. We will maintain several invariants:

- For any database in the support of the superposition, for any (x, y) pair where $x = \perp$, we have that $y = 0^n$. All $(\perp, 0^n)$ pairs are at the end of the list.
- For any database in the support of the superposition, if (x, y) occurs before (x', y'), it must be that $x < x'$.
- For any of the ℓ positions that have been specified, the y registers are in a state that is orthogonal to the uniform superposition $|\tau\rangle$ (indicating that in the Fourier domain, the registers do *not* contain 0).

We also need to describe several procedures on databases. Let $|D|$ be the number of pairs $(x, y) \in D$ for $x \neq \perp$. For a database D with $|D| < t$ and $D(x) = \perp$, write $D \cup (x, y)$ to be the new database obtained by adding the pair (x, y) to D, inserting in the appropriate spot to maintain the ordering of the x values. Since $|D|$ was originally less than t, there will be at least one $(\perp, 0^n)$ pair, which is deleted. Therefore, the overall number of pairs (including $\perp$s) in D and $D \cup \{(x, y)\}$ are the same.

Before describing how to process a query, we need to describe a local decompression procedure $\mathsf{StdDecomp}_x$ which acts on databases. This is a unitary operation. It suffices to describe its action on a set of orthonormal states. Let t be the current upper bound on the number of set points.

- For D such that $D(x) = \perp$ and $|D| < t$,

$$\mathsf{StdDecomp}_x |D\rangle = \frac{1}{\sqrt{2^n}} \sum_y |D \cup (x, y)\rangle$$

  That is, $\mathsf{StdDecomp}_x$ inserts into D the pair $(x, |\tau\rangle)$. This corresponds to decompressing the value of the database at position x

- For D such that $D(x) = \perp$ and $|D| = t$, $\mathsf{StdDecomp}_x|D\rangle = |D\rangle$. This means, if there is no room to expand for decompression, $\mathsf{StdDecomp}_x$ does nothing. Note that these states are illegal and $\mathsf{StdDecomp}_x$ will never by applied to such states.
- For a D' such that $D'(x) = \perp$ and $|D'| < t$,

$$\mathsf{StdDecomp}_x\left(\frac{1}{\sqrt{2^n}}\sum_y(-1)^{z\cdot y}|D' \cup (x,y)\rangle\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_y(-1)^{z\cdot y}|D' \cup (x,y)\rangle\rangle \text{ for } z \neq 0$$

$$\mathsf{StdDecomp}_x\left(\frac{1}{\sqrt{2^n}}\sum_y|D' \cup (x,y)\rangle\rangle\right) = |D'\rangle$$

  In other words, if D already is specified on x, and moreover if the corresponding y registers are in a state orthogonal to $|\tau\rangle$ (meaning they do not contain 0 in the Fourier domain), then there is no need to decompress and $\mathsf{StdDecomp}_x$ is the identity. On the other hand, if D is specified at x and the corresponding y registers are in the state $|\tau\rangle$, $\mathsf{StdDecomp}_x$ will remove x and the y register superposition from D.

Note that the left-hand sides of last two cases form an orthonormal basis for the span of $|D\rangle$ such that $D(x) \neq \perp$. The left-hand sides of the first two cases form an orthonormal basis for the remaining D. Thus, $\mathsf{StdDecomp}_x$ is defined on an orthonormal basis, which by linearity defines it on all states. The right-hand sides are the same basis states just in a different order. As such, this operation maps orthogonal states to orthogonal states, and is therefore unitary. Note that $\mathsf{StdDecomp}_x$ is actually an involution, as applying it twice results in the identity. Let $\mathsf{StdDecomp}$ be the related unitary operating on a quantum system over x, y, D states, defined by it's action on the computational basis states as:

$$|x,y\rangle \otimes |D\rangle = |x,y\rangle \otimes \mathsf{StdDecomp}_x|D\rangle$$

In other words, in superposition it applies $\mathsf{StdDecomp}_x$ to $|D\rangle$, where x is taken from the x registers.

For some additional notation, we will take $y \oplus \perp = y$ and $y \cdot \perp = 0$. Let $\mathsf{Increase}$ be theprocedure which initializes a new register $|(\perp, 0^n)\rangle$ and appends it to the end. In other words, $\mathsf{Increase}|x,y\rangle \otimes |D\rangle = |x,y\rangle \otimes |D\rangle|(\perp, 0^n)\rangle$, where $|D\rangle|(\perp, 0^n)\rangle$ is interpreted as a database computing the same partial function as D, but with the upper bound on number of points increased by 1.

Let $\mathsf{CStO}', \mathsf{CPhsO}'$ be unitaries defined on the computational basis states as

$$\mathsf{CStO}'|x,y\rangle \otimes |D\rangle = |x, y \oplus D(x)\rangle \otimes |D\rangle$$
$$\mathsf{CPhsO}'|x,y\rangle \otimes |D\rangle = (-1)^{y\cdot D(x)}|x,y\rangle \otimes |D\rangle$$

Finally, we describe the CStO and CPhsO oracles:

$$\mathsf{CStO} = \mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp} \circ \mathsf{Increase}$$
$$\mathsf{CPhsO} = \mathsf{StdDecomp} \circ \mathsf{CPhsO}' \circ \mathsf{StdDecomp} \circ \mathsf{Increase}$$

In other words, increase the bound on the number of specified points, then uncompress at x (which is ensured to have enough space since we increased the bound), apply the query (which is ensured to be specified since we decompressed), and then re-compress.

Lemma 4. CStO *and* StO *are perfectly indistinguishable.* CPhsO *and* PhO *are perfectly indistinguishable. That is, for any adversary A, we have* $\Pr[A^{\mathsf{CStO}}() = 1] = \Pr[A^{\mathsf{StO}}() = 1]$*, and for any adversary B, we have* $\Pr[B^{\mathsf{CPhsO}}() = 1] = \Pr[A^{\mathsf{PhO}}() = 1]$.

Proof. We prove the case for CStO and StO, the other case being almost identical. We prove security through a sequence of hybrids.

Hybrid 0. In this case, the adversary interacts with StO. That is, the oracle's database is initialized to the uniform superposition over all H, and each query performs the unitary mapping $|x, y\rangle \otimes |H\rangle \mapsto |x, y \oplus H(x)\rangle \otimes |H\rangle$.

Hybrid 1. In this hybrid, we use a slightly different way of representing the function H. Instead of writing H as a truth table, we represent it as a complete database $D = ((0, H(0)), (1, H(1)), \dots, (2^m - 1, H(2^m - 1)))$. Here, the upper bound on the number of determined points is exactly 2^m. The oracle's state starts out as

$$\frac{1}{\sqrt{2^{n2^m}}} \sum_H |((0, H(0)), (1, H(1)), \dots, (2^m - 1, H(2^m - 1)))\rangle$$

The update procedure for each query is simply CStO', meaning that each query maps $|x, y\rangle \otimes |((0, H(0)), (1, H(1)), \dots, (2^m - 1, H(2^m - 1)))\rangle$ to $|x, y \oplus H(x)\rangle \otimes |((0, H(0)), (1, H(1)), \dots, (2^m - 1, H(2^m - 1)))\rangle$.

Hybrid 1 is identical to **Hybrid 0**, except that we have inserted the input points $1, \dots, 2^m - 1$ into the oracle's state, which has no effect on the adversary.

Hybrid 2. Next, introduce a global decompression procedure $\mathsf{StdDecomp}'$, which applies $\mathsf{StdDecomp}_x$ for all x in the domain, one at a time from 0 up to $2^m - 1$.

We observe that when the upper bound on determined points is 2^m, then $\mathsf{StdDecomp}_x$ commutes with $\mathsf{StdDecomp}_{x'}$ for any x, x'. This readily follows from the fact that when the upper bound is $t = 2^m$, $D(x) = \bot$ implies $|D| < t$.

In **Hybrid 2**, the oracle starts out as the empty database with upper bound 2^m. Then, each query is implemented as $\mathsf{StdDecomp}' \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}'$.

Notice that $\mathsf{StdDecomp}'$ only affects the oracle's registers and therefore commutes with the any computation on the adversary's side. Also notice that between each two queries, $\mathsf{StdDecomp}'$ is applied twice and that it is an involution. Therefore the two applications cancel out. At the beginning, $\mathsf{StdDecomp}'$ is applied to an empty database, which maps it to the uniform superposition

$$\frac{1}{\sqrt{2^{n2^m}}} \sum_H |((0, H(0)), (1, H(1)), \dots, (2^m - 1, H(2^m - 1)))\rangle$$

before the first application of CStO'. Therefore, this hybrid is perfectly indistinguishable from **Hybrid 1**.

Hybrid 3. This hybrid applies $\mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}$ for each query.

To prove indistinguishability from **Hybrid 2**, consider a database D with upper bound 2^m but where $|D| = \ell$ for some $\ell \leq 2^m$. Notice that for any D' in the support of $\mathsf{StdDecomp}_{x'}|D\rangle$, $D'(x) = D(x)$ for all $x \neq x'$. This means

$$\begin{aligned}\mathsf{CStO}' \circ \mathsf{StdDecomp}_{x'}\left(|x, y\rangle \otimes |D\rangle\right) &= \mathsf{StdDecomp}_{x'}\left(|x, y \oplus D(x)\rangle \otimes |D\rangle\right)\\ &= \mathsf{StdDecomp}_{x'} \circ \mathsf{CStO}'(|x, y\rangle \otimes |D\rangle)\end{aligned}$$

In other words, when the query register contains $x \neq x'$, $\mathsf{StdDecomp}_{x'}$ and CStO' commute. Therefore,

$$\begin{aligned}\mathsf{StdDecomp}' \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}'(|x, y\rangle \otimes |D\rangle) &= \mathsf{StdDecomp}_x \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}_x(|x, y\rangle \otimes |D\rangle)\\ &= \mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}(|x, y\rangle \otimes |D\rangle)\end{aligned}$$

This shows that **Hybrid 2** and **Hybrid 3** are identical.

Hybrid 4. Finally, this hybrid is the compressed standard oracle: the oracle's state starts out empty, and CStO is applied for each query.

To prove equivalence, first notice that for any x, y, D, $\mathsf{StdDecomp} \circ \mathsf{CStO}' \circ \mathsf{StdDecomp}(|x, y\rangle \otimes |D\rangle)$ has support on databases D' such that $|D'| \leq |D| + 1$. Indeed, all D' are defined on the same inputs except for possibly the input x.

This means that after q queries in **Hybrid 3**, the oracle's registers only have support on D containing at most q defined points; the remaining $\geq 2^m - q$ points are all $(\bot, 0^n)$. Therefore, we can discard all but the first q pairs in D, without affecting the adversary's state. The result is identical to **Hybrid 4**. $\square$

In Appendix C, we give several more oracle variations; while not used in this work, they may be useful in other settings. These variations also provide an alternative way to arrive at the compressed standard oracles.

3.1 A Useful Lemma

Here, we provide a lemma which relates the adversary's knowledge of an oracle output to the probability that point appears in the compressed oracle database. This lemma is proved in Appendix B.3, and follows from a straightforward (albeit delicate) analysis off the action of CStO.

Lemma 5. *Consider a quantum algorithm A making queries to a random oracle H and outputting tuples $(x_1, \ldots, x_k, y_1, \ldots, y_k, z)$. Let R be a collection of such tuples. Suppose with probability p, A outputs a tuple such that (1) the tuple is in R and (2) $H(x_i) = y_i$ for all i. Now consider running A with the oracle* CStO*, and suppose the database D is measured after A produces its output. Let p' be the probability that (1) the tuple is in R, and (2) $D(x_i) = y_i$ for all i (and in particular $D(x_i) \neq \bot$). Then $\sqrt{p} \leq \sqrt{p'} + \sqrt{k/2^n}$*

4 Quantum Query Bounds Using Compressed Oracles

In this section, we re-prove several known query complexity lower bounds, as well as provide some new bounds. All these bounds follow from simple applications of our compressed oracles.

4.1 Optimality of Grover Search

Here, we re-prove that the quadratic speed-up of Grover search is optimal. Specifically, we prove that for a random function $H : \{0,1\}^m \to \{0,1\}^n$, any q query algorithm has a success probability of at most $O(q^2/2^n)$ for finding a pre-image of 0^n (or any fixed value).

Theorem 1. *For any adversary making q queries to* CStO *or* CPhsO *and an arbitrary number of database read queries, if the database D is measured after the q queries, the probability it contains a pair of the form $(x, 0^n)$ is at most $O(q^2/2^n)$.*

Proof. Let $0^n \in D$ mean that D contains a pair of the form $(x, 0^n)$. The compressed oracle's database starts out empty, so the probability $0^n \in D$ is zero. We will show that the probability cannot rise too much with each query. We consider compressed phase queries, CPhsO. Compressed standard queries are handled analogously. Consider the joint state of the adversary and oracle just before the qth CPhsO query:

$$|\psi\rangle = \sum_{x,y,z,D} \alpha_{x,y,z,D} |x, y, z\rangle \otimes |D\rangle$$

Where D represents the compressed phase oracle, x, y as the query registers, and z as the adversary's private storage. Define P as the projection onto the span of basis states $|x, y, z\rangle \otimes |D\rangle$ such that $0^n \in D$. Our goal will be to relate the norms of $P|\psi\rangle$ (the magnitude before the query) to $P \cdot \mathsf{CPhsO}|\psi\rangle$ (the magnitude after the query).

Define projections Q onto states such that (1) $0^n \notin D$ (meaning the database does not yet contain 0^n), (2) $y \neq 0$ (meaning CPhsO will affect D), and (3) $D(x) = \bot$ (meaning D has not yet been specified at x). Define projection R onto states such that $0^n \notin D$, $y \neq 0$ and $D(x) \neq \bot$; projection S onto states such that $0^n \notin D$, $y = 0$. Then $P + Q + R + S = \mathbf{I}$.

Consider $Q|\psi\rangle$. CPhsO maps basis states $|x, y, z\rangle \otimes |D\rangle$ in the support of $Q|\psi\rangle$ to $|x, y, z\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_w (-1)^{y \cdot w} |D \cup (x, w)\rangle$. Since $0^n \notin D$, applying P to this state will yield $|x, y, z\rangle \otimes \frac{1}{\sqrt{2^n}} |D \cup (x, 0^n)\rangle$. Notice that the images of the different basis states are orthogonal. Therefore, $\|P \cdot \mathsf{CPhsO} \cdot Q|\psi\rangle\| = \frac{1}{\sqrt{2^n}} \|Q|\psi\rangle\|$.

For basis vectors in the support of R, we must have $D(x) \notin \{\bot, 0^n\}$. Let D' be the database with x removed, and write $D = D' \cup (x, w)$ for $w = D(x)$. Then some algebraic manipulations show that $\mathsf{CPhsO}|x,y,z\rangle \otimes |D' \cup (x,w)\rangle$ is:

$$|x,y,z\rangle \otimes \left((-1)^{y\cdot w}\left(|D' \cup (x,w)\rangle + \frac{1}{\sqrt{2^n}}|D'\rangle\right) \right.$$
$$\left. + \frac{1}{2^n}\sum_{y'}(1-(-1)^{y\cdot w}-(-1)^{y\cdot y'})|D' \cup (x,y')\rangle \right)$$

Then $P \cdot \mathsf{CPhsO}|x,y,z\rangle \otimes |D' \cup (x,w)\rangle = \frac{-(-1)^{y\cdot w}}{2^n}|x,y,z\rangle \otimes |D' \cup (x,0^n)\rangle$. Write $R|\psi\rangle = \sum_{x,y,z,D',w}\alpha_{x,y,z,D',w}|x,y,z\rangle \otimes |D' \cup (x,w)\rangle$. Then $\|P\cdot\mathsf{CPhsO}\cdot R|\psi\rangle\|^2$ is equal to:

$$\frac{1}{4^n}\sum_{x,y,z,D'}\|\sum_w \alpha_{x,y,z,D',w}(-1)^{y\cdot w}\|^2 \le \frac{1}{2^n}\sum_{x,y,z,D'}\sum_w\|\alpha_{x,y,z,D',w}\|^2 = \frac{1}{2^n}\|R|\psi\rangle\|^2$$

Finally, $\|P\cdot\mathsf{CPhsO}\cdot P|\psi\rangle\| \le \|P|\psi\rangle\|$ and $\mathsf{CPhsO}\cdot S|\psi\rangle = S|\psi\rangle$. Putting it all together, we have that $\|P\cdot\mathsf{CPhsO}|\psi\rangle\| \le \|P|\psi\rangle\| + \frac{1}{\sqrt{2^n}}(\|Q|\psi\rangle\| + \|R|\psi\rangle\|) \le \|P|\psi\rangle\| + \frac{1}{\sqrt{2^n}}$.

Therefore, after q queries, we have that the projection onto D containing a zero has norm at most $q/\sqrt{2^n}$. Now, the probability the database in $|\psi\rangle$ contains a 0^n is just the square of this norm, which is at most $\frac{q^2}{2^n}$. □

The following is obtained by combining Theorem 1 with Lemma 5:

Corollary 1. *After making q quantum queries to a random oracle, the probability of finding a pre-image of 0^n is at most $O(q^2/2^n)$.*

Proof. We will assume the adversary always makes a final query on it's output x, and outputs $(x, H(x))$. This comes at the cost of at most 1 query, so it does not affect the asymptotic result. Then we can use the relation $R(x,y)$ which accepts if and only if $y = 0^n$. In the second experiment of Lemma 5, the only way for the adversary to win is to have the database contain a pre-image of 0^n. As such, Theorem 1 shows $p' = O(q^2/2^n)$. Then Lemma 5 shows that $p = O(q^2/2^n)$, which is exactly the probability the adversary outputs a pre-image of 0^n when interacting with the real random oracle.

4.2 Collision Lower Bound

Theorem 2. *For any adversary making q queries to CStO or CPhsO and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

Proof. The proof involves changing just a few lines of the proof of Theorem 1. We define P to project onto databases D containing a collision, and re-define Q, R, S accordingly. Write $Q|\psi\rangle = \sum_{x,y,z,D}\alpha_{x,y,z,D}|x,y,z\rangle \otimes |D\rangle$. Then

$$P\cdot\mathsf{CPhsO}\cdot Q|\psi\rangle = \sum_{x,y,z,D}\alpha_{x,y,z,D}|x,y,z\rangle \otimes \frac{1}{\sqrt{2^n}}\sum_{w\in D}|D \cup (x,w)\rangle$$

We can write this as the $\frac{1}{\sqrt{2^n}}\sum_i |\phi_i\rangle$, where $|\phi_i\rangle$ is the partial sum which sets w to be the ith element in D (provided it exists). The $|\phi_i\rangle$ are orthogonal, and satisfy $\||\phi_i\rangle\| \leq \|Q|\psi\rangle\|$. Moreover, after q queries D has size at most q, and so there are at most q of the $|\phi_i\rangle$. Therefore, $\|P \cdot \mathsf{CPhsO} \cdot Q|\psi\rangle\| \leq \sqrt{q/2^n}\|Q|\psi\rangle\|$.

By a similar argument, $\|P{\cdot}\mathsf{CPhsO}{\cdot}R|\psi\rangle\| \leq \sqrt{q/2^n}\|R|\psi\rangle\|$. Putting everything together, this shows that the norm of $P|\psi\rangle$ increases by at most $\sqrt{q/2^n}$ with each query. Therefore, after q queries, the total norm is at most $\sqrt{q^3/2^n}$, giving a probability of $q^3/2^n$. □

Corollary 2. *After making q quantum queries to a random oracle, the probability of finding a collision is at most $O(q^3/2^n)$.*

4.3 More General Settings

We can easily generalize even further. Let R be a relation on ℓ-tuples over $\{0,1\}^n$. Say that R is *satisfied* on a database D if D contains ℓ distinct pairs (x_i, y_i) such that $R(y_1, \ldots, y_\ell) = 1$. Let $k(q)$ be the maximum number of y that can be added to an unsatisfied database of size at most $q-1$ to make it satisfied.

Theorem 3. *For any adversary making q queries to CStO or CPhsO and an arbitrary number of database read queries, if the database D is measured after the q queries, the resulting database will be satisfied with probability at most $O(q^2 k(q)/2^n)$.*

For the k-sum problem, there are at most $\binom{q}{k-1}$ incomplete tuples that can be completed by adding a new point. As such, $k(q) \leq \binom{q}{k-1} \leq q^{k-1}$. This gives:

Corollary 3. *After making q quantum queries to a random oracle, the probability of finding k distinct inputs x_i such that $\sum_i H(x_i) = 0^n$ is at most $O(q^{k+1}/2^n)$.*

5 Indifferentiability of A Simple Domain Extender

5.1 Definitions

Let $h : \{0,1\}^m \to \{0,1\}^n$ be a random oracle, and let $C^h : \{0,1\}^M \to \{0,1\}^N$ be a polynomial-sized stateless classical circuit that makes oracle queries to h.

Definition 2. *Let $H : \{0,1\}^M \to \{0,1\}^N$ be a random function. A stateful quantum polynomial-time simulator $\mathsf{Sim}^H : \{0,1\}^m \to \{0,1\}^n$ is* indifferentiable *for C if, for any polynomial-time distinguisher $\mathcal{D}$ making queries to h, H,*

$$|\Pr[\mathcal{D}^{h,C^h}() = 1] - \Pr[\mathcal{D}^{\mathsf{Sim}^H,H}() = 1]| < \mathsf{negl}$$

Definition 3. *C^h is quantum indifferentiable from a random oracle if there exists an indifferentiable simulator Sim for C.*

Intuitively, in the "real" world, h is a random function and H is set to be C^h. C^h is indifferentiable if this real world is indistinguishable from an "ideal" world, where H is a random function, and h is set to be Sim^h for some efficient simulator Sim.

In order to help us prove indifferentiability of a simulator Sim, we introduce two weaker requirements. The first is *indistinguishability*, a weakened version of indifferentiability where the distinguisher is not allowed any queries to H:

Definition 4. *A simulator* Sim *is* indistinguishable *if, for any polynomial-time distinguisher $\mathcal{D}$ making queries to h,*

$$|\Pr[\mathcal{D}^h() = 1] - \Pr[\mathcal{D}^{\mathsf{Sim}^H}() = 1]| < \mathsf{negl}$$

Next, we introduce the notion of *consistency*. Here, we set h to be simulated by Sim^H, and we ask the adversary to distinguish honest evaluations of H from evaluations of C^h (where again h is still simulated by Sim^H).

Definition 5. *A simulator* Sim *is* consistent *if, for any polynomial-time distinguisher $\mathcal{D}$ making queries to h, H, if H is simulated by Sim^H, then*

$$|\Pr[\mathcal{D}^{\mathsf{Sim}^H, H}() = 1] - \Pr[\mathcal{D}^{\mathsf{Sim}^H, C^{\mathsf{Sim}^H}}() = 1]| < \mathsf{negl}$$

Lemma 6. *Any consistent and indistinguishable simulator is indifferentiable.*

The proof of Lemma 6 is straightforward, and proved in Appendix B.2.

Finally, it is straightforward to adapt the definitions and Lemma 6 to handle the case of many random compression functions $h_1, \ldots, h_\ell$. In this case, C makes queries to $h_1, \ldots, h_\ell$, $\mathcal{D}$ has quantum oracle access to $h_1, \ldots, h_\ell$ and H, while S makes quantum queries to H and simulates $h_1, \ldots, h_\ell$.

5.2 A Simple Domain Extender

We now consider a simple domain extender. Let $h_1 : \{0,1\}^m \to \{0,1\}^n, h_2 : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^n$ be two functions. Let $C^{h_1,h_2}(x_1, x_2) = h_2(h_1(x_1), x_2)$

Theorem 4. *If h_1, h_2 are random oracles, the simple domain extender C is indifferentiable from a random oracle.*

Coron et al. [CDMP05] show that the indifferentiability of C is sufficient to prove the indifferentiability of Merkle-Damgård for a particular choice of prefix-free encoding (see paper for details). That part of the paper translates immediately to the quantum setting, so Theorem 4 then shows quantum indifferentiability for the same prefix free encoding. In Section D, we show more generally that Merkle-Damgård is indifferentiable for *any* choice of prefix-free encoding. All the main ideas for the full proof are already contained in the proof of Theorem 4 below, just the details get a bit more complicated in the more general setting.

5.3 Our Simulator

Before describing our simulator, we need some terminology. For a database D of input/output pairs, a *collision* is two pairs $(x_1, y_1), (x_2, y_2) \in D, x_1 \neq x_2$ such that $y_1 = y_2$. For an input $(y, x_2) \in \{0,1\}^n \times \{0,1\}^\ell$, a *completion* in D is a pair $(x_1, y) \in D$. For such a completion, we will call $w = (x_1, x_2)$ the associated input.

We define a classical procedure $\mathsf{FindInput}$. $\mathsf{FindInput}$ takes as input $x \in \{0,1\}^n \times \{0,1\}^\ell$, and a database D. It parses x as $(y, x_2) \in \{0,1\}^n \times \{0,1\}^\ell$. Then, it looks for a completion $(x_1, y) \in D$. If found, it will take, say, the completion with the smallest x_1 value, and output $(b = 1, w = (x_1, x_2))$. If no completion is found, it will output $(b = 0, w = 0^{m+\ell})$. Note that for the output values in D, $\mathsf{FindInput}$ only needs to apply an equality check on those values, testing if they contain y. By applying such an equality check to each output register, it can compute b and w. Looking forward, when we implement $\mathsf{FindInput}$ in superposition, this means $\mathsf{FindInput}$ only touches the output registers of D by making a computational basis test.

We are now ready to describe our simulator. Sim will keep a (superposition over) database D_a, which represents the simulation of the random oracle h_a that it will update according to the CStO update procedure. D_a is originally empty. It will also have a private random oracle h_b. For concreteness, h_b will be implemented using another instance of CStO, but it will be notationally convenient to treat h_b as being a uniformly random function.

On h_1 queries, Sim makes a query to h_a, performing the appropriate CStO update procedure to D_a. On h_2 queries, Sim performs a unitary operation with the following action on basis states:

$$|x, y\rangle \otimes |D_a\rangle \mapsto \begin{cases} |x, y \oplus h_b(x)\rangle \otimes |D_a\rangle & \text{if } \mathsf{FindInput}(x, D_a) = (0, 0^{m+\ell}) \\ |x, y \oplus H(w)\rangle \otimes |D_a\rangle & \text{if } \mathsf{FindInput}(x, D_a) = (1, w) \end{cases}$$

This unitary is straightforward to implement with a single query to each of h_b and H, and is detailed in Appendix B.4.

In the next three subsections, we prove that our simulator is indifferentiable. In Section 5.4, we prove a useful commutativity lemma. Then in Sections 5.5 and 5.6, we prove the indistinguishability and consistency, respectively, of Sim. By Lemma 6, this proves that Sim is indifferentiable, proving Theorem 4.

5.4 The Almost Commutativity of StdDecomp and FindInput

Lemma 7. *Consider a quantum system over x, D, x', z. The following two unitaries $O(1/\sqrt{2^n})$-almost commute:*

- $\mathsf{StdDecomp}$*, acting on the x, D registers.*
- $\mathsf{FindInput}$*, taking as input the D, x' registers and XORing the output into z.*

The intuition is that, for $\mathsf{StdDecomp}$ to have any effect, either (1) $D(x) = \bot$ or (2) $D(x)$ is in uniform superposition; $\mathsf{StdDecomp}$ will simply toggle between the two cases. Now, a uniform superposition puts a weight of $1/\sqrt{2^n}$ on each possible

y value. Since there is only a single possible y value for $D(x)$ that matches x', it is exponentially unlikely that FindInput will find a match at input x in Case (2). On the other hand, it will *never* find a match at input x in Case (1). Hence, there is an exponentially small error between the action of FindInput on these two cases. We prove the lemma formally in Appendix B.5.

5.5 Indistinguishability

Lemma 8. Sim *is indistinguishable. In particular, for any distinguisher $\mathcal{D}$ making at most q queries to h_1, h_2,*

$$|\Pr[\mathcal{D}^{h_1,h_2}() = 1] - \Pr[\mathcal{D}^{\mathsf{Sim}^H}() = 1]| < O(q^2/\sqrt{2^n})$$

Proof. Recall that in the ideal world where h_1, h_2 are simulated by Sim^H, h_1 is implemented by a CStO oracle on database D_a. By applying Lemma 4, we can think of the simulator's other oracle h_b as another instance of CStO for a database D_b. Additionally, H can be simulated with yet another instance of CStO for a database E. Similarly, in the real world, h_1, h_2 will be implemented by independent instances of CStO with databases D_a, D_b. Note that, in either case, h_1 is implemented by a CStO oracle on database D_a. Therefore, the only difference between the two cases is how h_2 is implemented.

We define a classical encoding procedure Encode for pairs D_a, D_b of databases. Intuitively, Encode will scan the values $((z, x_2), y)$ in D_b, seeing if any of the (z, x_2) values correspond to a completion in D_a. If so, such a completion will have an associated input w. Encode will reasonably guess that such a completion corresponds to an evaluation of $H(w) = C^{h_1,h_2}(w)$. Therefore, Encode will remove the value $((z, x_2), y)$ in D_b, and add the pair (w, y) to a new database E, intuitively representing the oracle H. In more detail, Encode does the following:

- For each pair $((z, x_2), y) \in D_b$, run $\mathsf{FindInput}((z, x_2), D_a) = (b, w)$. If $b = 1$, re-label the pair to (w, y)
- Remove all re-labeled pairs D_b (which are easily identifiable since the input will be larger) and place them in a new database E.

We define the following Decode procedure, which operates on triples D_a, D_b, E:

- Merge the databases D_b, E
- For each pair (w, y) that was previously in E, where $w = (x_1, x_2)$, evaluate $z = D_a(x_1)$. Re-label (w, y) to $((z, x_2), y)$. If $z = \perp$ or if the input (z, x_2) was already in the database, output $\perp$ and abort.

Note that Encode, Decode are independent of the order elements are processed. It also follows immediately from the descriptions above that $\mathsf{Decode}(\mathsf{Encode}(D_a, D_b)) = (D_a, D_b)$. Therefore, Encode can be implemented in superposition, giving the unitary that maps $|D_a, D_b\rangle$ to $|\mathsf{Encode}(D_a, D_b)\rangle$. Also note that $\mathsf{Encode}(\emptyset, \emptyset) = (\emptyset, \emptyset, \emptyset)$.

With this notation in hand, we are now ready to prove security: consider a potential distinguisher $\mathcal{D}$. We prove security through a sequence of hybrids.

Hybrid 0. This is the real world, where h_1, h_2 are random oracles. Let p_0 be the probability $\mathcal{D}$ outputs 1 in this case.

Hybrid 1. This is still the real world, but we add an abort condition. Namely, after any query to h_1, we measure if the database h_a contains a collision; if so, we immediately abort and stop the simulation. Let p_1 be the probability $\mathcal{D}$ outputs 1 in Hybrid 1.

Lemma 9. $|p_1 - p_0| \leq O(\sqrt{q^3/2^n})$

Proof. First, suppose that before the ith query to h_1, the superposition over h_a has support only on databases containing no collisions. Let $|\psi\rangle$ be the joint state of the adversary and simulator just after the query to h_1. Then write $|\psi\rangle = |\psi_0\rangle + |\psi_1\rangle$ where $|\psi_0\rangle$ is the projection onto states where h_a has no collisions, and $|\psi_1\rangle$ is the projection onto states where h_a contains at least one collision. Following the proof of Theorem 2, we know that $\||\psi_1\rangle\| \leq \sqrt{i/2^n}$.

Therefore, if we let $|\psi_q\rangle$ be the joint state after the qth query in **Hybrid 0** and $|\phi_q\rangle$ the joint state in **Hybrid 2**, we would have that $\||\psi_q\rangle - |\phi_q\rangle\| \leq \sum_{i=0}^{q} \sqrt{i/2^n} \leq O(\sqrt{q^3/2^n})$. By Lemma 1, this means that $|p_1 - p_0| \leq O(\sqrt{q^3/2^n})$ as desired. □

Hybrid 2. In this hybrid, there are three databases D_a, D_b, E, initialized to $|\emptyset, \emptyset, \emptyset\rangle$. Each query is answered in the following way:

– Apply Decode to the D_a, D_b, E registers. Measure if Decode gives $\perp$, in which case abort. Otherwise, there are now just two database registers D_a, D_b.
– Answer an h_1 (resp. h_2) query by applying the CStO update procedure to D_a (resp. D_b).
– Apply Encode to D_a, D_b.
– Apply the collision check to the database D_a.

Let p_2 be the probability $\mathcal{D}$ outputs 1 in Hybrid 2.

Lemma 10. $p_1 = p_0$

Proof. We start with **Hybrid 1**. First, by Lemma 4, we can implement D_a, D_b in Hybrid 1 as independent instances of CStO. Now, between all the queries insert Encode followed by Decode. Also insert the two procedures before the first query. Now each query is preceded by a Decode and followed by a collision check and an Encode. Note that Encode, Decode do not affect the database D_a, and so commute with the collision check. Therefore, we can swap the order of the collision check and Encode that follow each query.

By merging the Decode, query, Encode and collision check operations together, we get exactly the update procedure of Hybrid 2. All that's left is an initial Encode procedure at the very beginning, which produces $|\emptyset, \emptyset, \emptyset\rangle$ as the database state, just as in Hybrid 2. □

Hybrid 3. This hybrid is the ideal world, where h_1, h_2 queries are answered by Sim, except that we will have the abort condition if a collision in h_a is ever found. In other words, instead of decoding, applying the query, and then encoding, in Hybrid 3 we act directly on the encoded state using the algorithms specified by Sim. For h_1 queries, the difference from Hybrid 2 is just that the queries are made directly to h_a, instead of Decode, then h_a query, then Encode. For h_2 queries, the differences appear more substantial. h_2 queries, on superpositions over x, y, D_a, D_b, E, can be summarized as follows:

1. Compute the unitary mapping $|x, y, D_a, D_b, E\rangle \mapsto |x, y, D_a, D_b, E, (b, w) = \mathsf{FindInput}(x, D_a)\rangle$
2. In superposition, apply the following conditional procedures:
3. Conditioned on $b = 0$,
   (a) Apply $\mathsf{StdDecomp}$ to uncompress D_b at x.
   (b) Apply in superposition the map $|x, y, D_a, D_b, E, b, w\rangle \mapsto |x, y \oplus D_b(x), D_a, D_b, E, b, w\rangle$.
   (c) Apply $\mathsf{StdDecomp}$ to re-compress D_b at x.
4. Conditioned on $b = 1$,
   (a) Apply $\mathsf{StdDecomp}$ to uncompress E at w.
   (b) Apply in superposition the map $|x, y, D_a, D_b, E, b, w\rangle \mapsto |x, y \oplus E(w), D_a, D_b, E, b, w\rangle$.
   (c) Apply $\mathsf{StdDecomp}$ to re-compress E at w.
5. Uncompute (b, w) by running $\mathsf{FindInput}(x, D_a)$ in superposition again.

Let p_3 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 11. $|p_3 - p_2| \leq O(q^2/\sqrt{2^n})$.

Proof. We start with the very last query, and gradually change the queries one-by-one from how they were answered in Hybrid 2 to Hybrid 3.

For h_1 queries, we observe that it suffices to swap the order of Encode and CStO. Indeed, suppose we move the final Encode to come before CStO. The previous query ended with an Encode, and now the current query begins with Decode then Encode. Since $\mathsf{Decode} \circ \mathsf{Encode}$ is the identity, all thee of these operations collapse into a single Encode, which we keep at the end of the previous query. The result is that the current query is just a direct call to CStO, as in Hybrid 3. Then it remains to show that we can swap the order of Encode and CStO. For this, notice that Encode only interacts with D_a through $\mathsf{FindInput}$. As such, all steps in $\mathsf{Encode}, \mathsf{CStO}$ commute except for the two $\mathsf{StdDecomp}$ operations in CStO and the $\mathsf{FindInput}$ operation in Encode for each entry in D_b (plus another $\mathsf{FindInput}$ operation when un-computing the scratch-space of Encode in order to implement in superposition). By Lemma 7, these $\leq 4q$ operations each $O(1/\sqrt{2^n})$-almost commute, meaning Encode and CStO $O(q/\sqrt{2^n})$-almost commute.

For h_2 queries, fix an x, D_a and suppose D_a contains no collisions as guaranteed. There are two cases:

– $\mathsf{FindInput}(x, D_a) = (0, 0^{m+\ell})$. Then in Hybrid 2, decoding/encoding does not affect the labeling for an (x, z) pair in D_b. As such, Hybrid 2 will uncompress D_b at x, apply the map $|x, y, D_a, D_b, E\rangle \mapsto |x, y \oplus D_b(x), D_a, D_b, E\rangle$ and then re-compress D_b at x, for these x, D_a.

– $\mathsf{FindInput}(x, D_a) = (1, w)$. Then in Hybrid 2, by the collision-freeness of D_a, decoding will re-label a $(w, z) \in E$ (if present) to $(x, z) \in D_b$. The effect of Hybrid 2 in this case will be to uncompress E at w, apply the map $|x, y, D_a, D_b, E\rangle \mapsto |x, y \oplus E(x), D_a, D_b, E\rangle$, and then re-compress E at w.

In either case, answering h_2 queries in Hybrid 2 and 3 act identically. Therefore, this change introduces no error.

After q h_1 or h_2 queries, the total error between Hybrid 1 and Hybrid 2 is at most $O(q^2/\sqrt{2^n})$. □

Hybrid 4. This is the ideal world, where we remove the abort condition from Hybrid 3. Let p_4 be the probability $\mathcal{D}$ outputs 1 in Hybrid 4. By an almost identical proof to that of Lemma 9, we have:

Lemma 12. $|p_4 - p_3| \leq O(\sqrt{q^3/2^n})$

Summing up, we have that $|p_0 - p_4| < O(q^2/\sqrt{2^n})$, proving Lemma 8. □

5.6 Consistency

Lemma 13. Sim *is consistent. In particular, for any distinguisher $\mathcal{D}$ making at most q quantum queries to h_1, h_2, H,*

$$|\Pr[\mathcal{D}^{\mathsf{Sim}^H, H}() = 1] - \Pr[\mathcal{D}^{\mathsf{Sim}^H, C^{\mathsf{Sim}^H}}() = 1]| < O(\sqrt{q^3/2^n})$$

In other words, h_1, h_2 are simulated as Sim^H, and the adversary cannot distinguish between H and C^{h_1,h_2}.

Proof. We first work out how H queries are answered using C^{h_1,h_2}, when we simulate h_1, h_2 using Sim^H. The input registers will be labeled with $x = (x_1, x_2)$, and the output registers labeled with y.

1. First, make an h_1 query on the x_1 registers, writing the output to some new registers initialized to $z = 0^n$. Since we are implementing h_1 using CStO, this is accomplished using the following steps:
   (a) Apply $\mathsf{StdDecomp}$ to un-compress D_a at x_1
   (b) Evaluate the map $|x_1, z, x_2, y\rangle \otimes |D_a\rangle \mapsto |x_1, z \oplus D_a(x_1), x_2, y\rangle \otimes |D_a\rangle$, where z is the new register that was initialized to 0.
   (c) Re-compress D_a at x_1 by applying $\mathsf{StdDecomp}$ again.
2. Next, make an h_2 query on input (z, x_2) (where z where the registers created previously) with output registers y. This has the effect of mapping to:
$$|x_1, z, x_2, y \oplus h_b(x)\rangle \otimes |D_a\rangle \text{ if } \mathsf{FindInput}((z, x_2), D_a) = (0, 0^{m+\ell})$$
$$|x_1, z, x_2, y \oplus H(w)\rangle \otimes |D_a\rangle \text{ if } \mathsf{FindInput}(z, x_2), D_a) = (1, w)$$
3. Finally, make another h_1 query to un-compute the value of z. This is accomplished in the following steps:
   (a) Apply $\mathsf{StdDecomp}$ to un-compress D_a at x_1
   (b) Evaluate the map $|x_1, z, x_2, y\rangle \otimes |D_a\rangle \mapsto |x_1, z \oplus D_a(x_1), x_2, y\rangle \otimes |D_a\rangle$.
   (c) Re-compress D_a at x_1 by applying $\mathsf{StdDecomp}$ again.
   (d) Then discard the z registers.

Let $\mathcal{D}$ be a potential distinguisher. We consider the following hybrids:

Hybrid 0. In this hybrid, H queries are answered using C^{h_1,h_2}, as worked out above. Let p_0 be the probability $\mathcal{D}$ outputs 1.

Hybrid 1. This hybrid is identical to Hybrid 0, except that Steps 1c and 3a are removed. Let p_1 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 14. $|p_1 - p_0| < O(q/\sqrt{2^n})$.

Proof. Since Steps 1c and 3a are inverses of each other, Hybrid 1 is equivalent to moving Step 3a up to occur just after Step 1c. Note that Step 2 only interacts with D_a through two applications of $\mathsf{FindInput}$ (one for computing, one for un-computing), which in turn $O(1/\sqrt{2^n})$-almost commutes with Step 1c. By Lemma 7, each query to H therefore creates an error $O(1/\sqrt{2^n})$, yielding a total error of $O(q/\sqrt{2^n})$. □

Hybrid 2. This hybrid is identical to Hybrid 2, except that after each query we measure if the database D_a contains a collision. If so, we abort and stop the simulation. Let p_2 be the probability $\mathcal{D}$ outputs 1 in this hybrid. By an almost identical proof to that of Lemma 9, we have:

Lemma 15. $|p_2 - p_1| < O(\sqrt{q^3/2^n})$

Hybrid 3. This hybrid is identical to Hybrid 2 as outlined above, except that:

– Steps 1c and 3a are removed (as in Hybrid 1 and 2)
– The operation in Step 2 is replaced with

$$|x_1, z, x_2, y\rangle \otimes |D_a\rangle \mapsto |x_1, z, x_2, y \oplus H(x_1, x_2)\rangle \otimes |D_a\rangle$$

In other words Hybrid 3 is identical to Hybrid 2, except that we change Step 2. Let p_3 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 16. $p_3 = p_2$.

Proof. In either hybrid, since we do not apply the Steps 1c and 3a, D_a is guaranteed to contain the pair (x_1, z), where z is the same as in Step 2. Therefore, in Hybrid 2, $\mathsf{FindInput}((z, x_2), D_a)$ is guaranteed to find a completion. Moreover, for D_a that contain no collisions, $\mathsf{FindInput}((z, x_2), D_a)$ will find exactly the completion (x_1, z). In this case, $w = (x_1, x_2)$, and Hybrid 2 will make a query to H on (x_1, x_2). The end result is that for D_a containing no collisions, Step 2 is identical in both Hybrids. Since the collision check guarantees no collisions in D_a, this shows that the two hybrids are identical. □

Hybrid 4. In this hybrid, H queries are made directly to H, but we still have the abort condition. Let p_4 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 17. $p_4 = p_3$

Proof. In Hybrid 3, what remains of Steps 1 and 3 are exact inverses of each other and moreover commute with the new Step 2 from Hybrid 3. Therefore, we can remove Steps 1 and 3 altogether without affecting how oracle queries are answered. The result is identical to Hybrid 4. □

Hybrid 5. This hybrid has H queries made directly to H, but without the abort condition. Let p_5 be the probability $\mathcal{D}$ outputs 1 in this hybrid. By an almost identical proof to that of Lemma 9, we have:

Lemma 18. $|p_5 - p_4| < O(\sqrt{q^3/2^n})$

Overall then $|p_0 - p_5| < O(\sqrt{q^3/2^n})$, finishing the proof of Lemma 13. $\square$

6 Fujisaki Okamoto CCA-Secure Encryption

Here, we summarize our results on the Fujisaki-Okamoto transformation [FO99]. The transformation starts with a symmetric key encryption scheme $(\mathsf{Enc}_S, \mathsf{Dec}_S)$ and a public key encryption scheme $(\mathsf{Gen}_P, \mathsf{Enc}_P, \mathsf{Dec}_P)$. Assuming only mild security properties of these two schemes (which are much easier to obtain than strong CCA security), the conversion produces a new public key scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ which is secure against chosen ciphertext attacks. Let G, H are two random oracles, where G outputs keys for Enc_S and H outputs the random coins used by Enc_P. The scheme is as follows:

- $\mathsf{Gen} = \mathsf{Gen}_P$.
- $\mathsf{Enc}(\mathsf{pk}, m)$ chooses a random $\delta \in \{0,1\}^n$, and computes $d \leftarrow \mathsf{Enc}_S(H(\delta), m)$. Then it computes $c \leftarrow \mathsf{Enc}_P(\mathsf{pk}, \delta; G(\delta, d))$, and outputs (c, d)
- $\mathsf{Dec}(\mathsf{sk}, (c, d))$ first computes $\delta' \leftarrow \mathsf{Dec}_P(\mathsf{sk}, c)$. Then it checks that $\mathsf{Enc}_P(\mathsf{pk}, \delta'; G(\delta', d)) = c$; if not, output $\perp$. Finally it computes and outputs $m' \leftarrow \mathsf{Dec}_S(H(\delta'), d)$

The main difficulty in the classical proof of security is allowing the reduction to answer decryption queries. The key idea is that, in order for the adversary to generate a valid ciphertext, it must have queried the oracles on δ. The reduction will simulate G, H on the fly by keeping track of tables of input/output pairs. When a chosen ciphertext query comes in, it will scan the tables looking for a δ that "explains" the ciphertext.

In the quantum setting, we run into a similar recording barrier as in the indifferentiability setting. Our key observation is that the output values of the G, H tables are only used for set membership tests. Just like equality tests used in our indifferentiability simulator, set membership tests in the primal and Fourier domain very nearly commute. As such, we can use our compressed oracles to mimic the classical proof following our techniques. Our reduction can even handle chosen ciphertext queries on quantum superpositions of ciphertexts. In Appendix F, we prove the following theorem:

Theorem 5. *If* $(\mathsf{Enc}_S, \mathsf{Dec}_S)$ *is one-time secure and* $(\mathsf{Gen}, \mathsf{Enc}_P, \mathsf{Dec}_P)$ *is well-spread and one-way secure, then* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is quantum CCA secure in the quantum random oracle model.*

Bibliography

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

[ATTU16] Mayuresh Vivekanand Anand, Ehsan Ebrahimi Targhi, Gelo Noel Tabia, and Dominique Unruh. Post-quantum security of the CBC, CFB, OFB, CTR, and XTS modes of operation. Cryptology ePrint Archive, Report 2016/197, 2016. `http://eprint.iacr.org/2016/197`.

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

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

[BES17] Marko Balogh, Edward Eaton, and Fang Song. Quantum collision-finding in non-uniform random functions. Cryptology ePrint Archive, Report 2017/688, 2017. `http://eprint.iacr.org/2017/688`.

[BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In *LATIN: : Theoretical Informatics, Latin American Symposium*, pages 163–169. Springer, Heidelberg, 1998.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

[BS13] Aleksandrs Belovs and Robert Spalek. Adversary lower bound for the k-sum problem. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 323–328, New York, NY, USA, 2013. ACM.

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

[CBH$^+$17] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. Post-quantum security of the sponge construction. Cryptology ePrint Archive, Report 2017/771, 2017. `http://eprint.iacr.org/2017/771`.

[CDG$^+$15] Daniel Cabarcas, Denise Demirel, Florian Göpfert, Jean Lancrenon, and Thomas Wunderer. An unconditionally hiding and long-term binding post-quantum commitment scheme. Cryptology ePrint Archive, Report 2015/628, 2015. `http://eprint.iacr.org/2015/628`.

[CDG+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. Cryptology ePrint Archive, Report 2017/279, 2017. http://eprint.iacr.org/2017/279.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Heidelberg, August 2005.

[DFG13] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat-Shamir transformation in a quantum world. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 62–81. Springer, Heidelberg, December 2013.

[Eat17] Edward Eaton. Leighton-micali hash-based signatures in the quantum random-oracle model. In *24th Annual Conference on Selected Areas in Cryptography (SAC2017)*, 2017. https://eprint.iacr.org/2017/607.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th ACM STOC*, pages 212–219. ACM Press, May 1996.

[IBM17] IBM. Ibm announces advances to ibm quantum systems and ecosystem, 2017. https://www-03.ibm.com/press/us/en/pressrelease/53374.wss.

[JZC+18] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. Ind-cca-secure key encapsulation mechanism in the quantum random oracle model, revisited. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, pages 96–125, Cham, 2018. Springer International Publishing.

[Mit14] Arno Mittelbach. Salvaging indifferentiability in a multi-stage setting. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 603–621. Springer, Heidelberg, May 2014.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 21–39. Springer, Heidelberg, February 2004.

[NIS17] NIST. Candidate quantum-resistant cryptographic algorithms publicly available, 2017. https://www.nist.gov/news-events/news/2017/12/candidate-quantum-resistant-cryptographic-algorithms-publicly-available.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework.

In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Heidelberg, May 2011.

[Son14] Fang Song. A note on quantum security for post-quantum cryptography. Cryptology ePrint Archive, Report 2014/709, 2014. http://eprint.iacr.org/2014/709.

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

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 755–784. Springer, Heidelberg, April 2015.

[Unr16] Dominique Unruh. Collapse-binding quantum commitments without random oracles. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 166–195. Springer, Heidelberg, December 2016.

[YAJ$^+$17] Youngho Yoo, Reza Azarderakhsh, Amir Jalali, David Jao, and Vladimir Soukharev. A post-quantum digital signature scheme based on supersingular isogenies. Cryptology ePrint Archive, Report 2017/186, 2017. http://eprint.iacr.org/2017/186.

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

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

[Zha15] Mark Zhandry. A note on the quantum collision and set equality problems. *Quantum Information and Computation*, 15(7& 8), 2015.

A Quantum Background

A quantum system Q is defined over a finite set B of classical states. We will generally consider $B = \{0,1\}^n$. A **pure** state over Q is an L_2-normalized vector in $\mathbb{C}^{|B|}$, which assigns a (complex) weight to each element in B. Thus the set of pure states forms a complex Hilbert space. A **qubit** is a quantum system defined over $B = \{0,1\}$. Given a quantum system Q_0 over B_0 and a quantum system Q_1 over B_1, we can define the product system $Q = Q_0 \times Q_1$ over $B = B_0 \times B_1 = \{(b_0, b_1) : b_0 \in B_0, b_1 \in B_1\}$. Given a state $v_0 \in Q_0$ and $v_1 \in Q_1$, we define the product state $v_0 \otimes v_1$ in the natural way. An n-qubit system is then $Q = Q_0^{\oplus n}$ where Q_0 is a single qubit.

Bra-ket notation. We will think of pure states as column vectors. The pure state that assigns weight 1 to x and weight 0 to each $y \neq x$ is denoted $|x\rangle$. The set $\{|x\rangle\}$ therefore gives an orthonormal basis for the Hilbert space of pure states. We will call this basis the "computational basis." If a state $|\phi\rangle$ is a linear combination of several $|x\rangle$, we say that $|\phi\rangle$ is in "superposition." For a pure state $|\phi\rangle$, we will denote the conjugate transpose as the row vector $\langle\phi|$.

Entanglement. In general, a pure state $|\phi\rangle$ over $Q_0 \times Q_1$ cannot be expressed as a product state $|\phi_0\rangle \otimes |\phi_1\rangle$ where $|\phi_b\rangle \in Q_b$. If $|\phi\rangle$ is not a product state, we say that the systems Q_0, Q_1 are **entangled**. If $|\phi\rangle$ is a product state, we say the systems are **un-entangled**.

Evolution of quantum systems. A pure state $|\phi\rangle$ can be manipulated by performing a unitary transformation U to the state $|\phi\rangle$. We will denote the resulting state as $|\phi'\rangle = U|\phi\rangle$.

Basic Measurements. A pure state $|\phi\rangle$ can be measured; the measurement outputs the value x with probability $|\langle x|\phi\rangle|^2$. The normalization of $|\phi\rangle$ ensures that the distribution over x is indeed a probability distribution. After measurement, the state "collapses" to the state $|x\rangle$. Notice that subsequent measurements will always output x, and the state will always stay $|x\rangle$.

If $Q = Q_0 \times Q_1$, we can perform a **partial measurement** in the system Q_0 or Q_1. If $|\phi\rangle = \sum_{x\in B_0, y\in B_1} \alpha_{x,y}|x,y\rangle$, partially measuring in Q_0 will give x with probability $p_x = \sum_{y\in B_1} |\alpha_{x,y}|^2$. $|\phi\rangle$ will then collapse to the state $\sum_{y\in B_1} \frac{\alpha_{x,y}}{\sqrt{p_x}}|x,y\rangle$. In other words, the new state has support only on pairs of the form (x,y) where x was the output of the measurement, and the weight on each pair is proportional to the original weight in $|\phi\rangle$. Notice that subsequent partial measurements over Q_0 will always output x, and will leave the state unchanged.

The above corresponds to measurement in the computational basis. Measurements in other bases are possible to, and defined analogously. We will generally only consider measurements in the computational basis; measurements in other bases can be implemented by composing unitary operations with measurements in the computational basis.

Efficient Computation. A quantum computer will be able to perform a fixed, finite set G of unitary transformations, which we will call **gates**. For concreteness, we will use so-called Hadamard, phase, CNOT and $\pi/8$ gates, but the precise choice is not important for this work, so long as the gate set is "universal" for quantum computing.

Let Q be a quantum system on n qubits. Each gate costs unit time to apply, and each partial measurement also costs unit time. Therefore, an efficient quantum algorithm will be able to make a polynomial-length sequence of operations, where each operation is either a gate from G or a partial measurement in the computational basis. Here, "polynomial" will generally mean polynomial in n.

Examples of Quantum Computations.

- **Quantum Fourier Transform.** Let Q_0 be a quantum system over $B = \mathbb{Z}_q$ for some integer q. Let $Q = Q_0^{\otimes n}$. The Quantum Fourier Transform (QFT) performs the following operation efficiently:

$$\mathsf{QFT}|x\rangle = \frac{1}{\sqrt{q^n}}\omega_q^{x\cdot y}\sum_{y\in\{0,1\}^n}|y\rangle$$

  where $\omega_q = e^{2\pi i/q}$.
  In this paper, we will always consider $q = 2$, so that $\omega_q = (-1)$.
- **Efficient Classical Computations.** Any function that can be computed efficiently classically can be computed efficiently on a quantum computer. More specifically, if f is computable by a polynomial-sized circuit, then there is a efficiently computable unitary U_f on the quantum system $Q = Q_{in}\otimes Q_{out}\otimes Q_{work}$ with the property that: $U_f|x,y,0\rangle = |x,y+f(x),0\rangle$.
  Here, Q_{in} is a quantum system over the set of possible inputs, Q_{out} is a quantum system over the set of possible outputs, and Q_{work} is another quantum system that is just used for workspace, and is reset after use.

Mixed states. A quantum system may, for example, be in a pure state $|\phi\rangle$ with probability 1/2, and a different pure state $|\psi\rangle$ with probability 1/2. This can occur, for example, if a partial measurement is performed on a product system.

This probability distribution on pure states cannot be described by a pure state alone. Instead, we say that the system is in a **mixed state**. The statistical behavior of a mixed state can be captured by **density matrix**. If the system is in pure state $|\phi_i\rangle$ with probability p_i, then the density matrix for the system is defined as $\rho = \sum_i p_i|\phi_i\rangle\langle\phi_i|$.

The density matrix is therefore a positive semi-definite complex Hermitian matrix with rows and columns indexed by the elements of B. The density matrix for a pure state $|\phi\rangle$ is given by the rank-1 matrix $|\phi\rangle\langle\phi|$. Any probability distribution over classical states can also be represented as a density matrix, namely the diagonal matrix where the diagonal entries are the probability values.

B Some Missing Details and Proofs

B.1 Proof of Lemma 1

Here, we prove Lemma 1, which we have reproduced below:

Lemma 1. *Let $|\phi\rangle$ and $|\psi\rangle$ be quantum states with Euclidean distance ϵ. Then the trace distance between $|\phi\rangle\langle\phi|$ and $|\psi\rangle\langle\psi|$ is $\epsilon\sqrt{1-\epsilon^2/4}\leq\epsilon$.*

Proof. Let $|\tau\rangle = (|\phi\rangle+|\psi\rangle)/2$ and $|\gamma\rangle = (|\phi\rangle-|\psi\rangle)/2$. Then $\||\gamma\rangle\| = \epsilon/2$ and by simple geometry $\||\tau\rangle\| = \sqrt{1-\epsilon^2/4}$. Moreover, $|\tau\rangle$ and $|\gamma\rangle$ are orthogonal. Now, we can write

$$|\phi\rangle\langle\phi|-|\psi\rangle\langle\psi| = (|\tau\rangle+|\gamma\rangle)(\langle\tau|+\langle\gamma|)-(|\tau\rangle-|\gamma\rangle)(\langle\tau|-\langle\gamma|) = 2(|\tau\rangle\langle\gamma|+|\gamma\rangle\langle\tau|)$$

This matrix is rank 2 and in the span of $|\tau\rangle$ and $|\gamma\rangle$. Writing the matrix in the orthonormal basis $\{(1/\sqrt{1-\epsilon^2/4})|\tau\rangle, (2/\epsilon)|\gamma\rangle\}$, we see that the matrix is equal to $\begin{pmatrix} 0 & \epsilon\sqrt{1-\epsilon^2/4} \\ \epsilon\sqrt{1-\epsilon^2/4} & 0 \end{pmatrix}$. This matrix has eigenvalues $\pm\epsilon\sqrt{1-\epsilon^2/4}$. Therefore, the trace norm is $\epsilon\sqrt{1-\epsilon^2/4}$.

B.2 Proof of Lemma 6

Here, we prove Lemma 6, which we have reproduced below:

Lemma 5. *Any consistent and indistinguishable simulator is indifferentiable.*

Proof. Consider a distinguisher $\mathcal{D}$. Consider the following hybrid sequence:

Hybrid 0. This hybrid is the "real" world where h is a random oracle, and $H = C^h$. Let p_0 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Hybrid 1. In this hybrid, h is simulated by Sim^H for a random oracle H. However, instead of answering H queries using H, we instead answer H queries by running C^h, where h are again simulated by Sim^H. Let p_1 be the probability $\mathcal{D}$ outputs 1 in this hybrid. Let $\mathcal{E}^h = \mathcal{D}^{h,C^h}$. Then E is an indistinguishability adversary with distinguishing advantage is exactly $|p_0 - p_1|$. Therefore $|p_0 - p_1| < \mathsf{negl}$.

Hybrid 2. Finally, this hybrid is the "simulated" world where H is a random oracle and h is simulated by Sim^H. Let p_2 be the probability $\mathcal{D}$ outputs 1 in this hybrid. By the consistency of Sim, $|p_1 - p_2| < \mathsf{negl}$.

Overall, then $|p_0 - p_2| < \mathsf{negl}$, as desired. □

B.3 Proof of Lemma 5

Here, we prove Lemma 5, which we have reproduced below:

Lemma 5. *Consider a quantum algorithm A making queries to a random oracle H and outputting tuples $(x_1, \dots, x_k, y_1, \dots, y_k, z)$. Let R be a collection of such tuples. Suppose with probability p, A outputs a tuple such that (1) the tuple is in R and (2) $H(x_i) = y_i$ for all i. Now consider running A with the oracle* CStO*, and suppose the database D is measured after A produces its output. Let p' be the probability that (1) the tuple is in R, and (2) $D(x_i) = y_i$ for all i (and in particular $D(x_i) \neq \bot$). Then*

$$\sqrt{p} \le \sqrt{p'} + \sqrt{k/2^n}$$

Proof. We first prove the case $k = 1$. Consider A interacting with CStO. We will write the final state of the adversary and oracle as:

$$\sum_{x,y,z,D} \alpha_{x,y,z,D,0}|x,y,z,D\rangle + \sum_{r\neq 0^n, x,y,z,D} \alpha_{x,y,z,D,r}\frac{1}{\sqrt{2^n}}\sum_{y'}(-1)^{y'\cdot r}|x,y,z,D\cup(x,y')\rangle$$

Here, the sum is over D such that $D(x) = \bot$ and z is some auxiliary work space of the adversary. Notice that the guarantee of CStO that the oracle's outputs do not contain 0^n in the Fourier domain mean that the sum on the right is only over $r \neq 0^n$. By normalization, we have that $\sum_{x,y,z,D,r} \|\alpha_{x,y,z,D,r}\|^2 = 1$ (here, we include the $r = 0^n$ case). The assumptions of the lemma statement imply that

$$p' = \frac{1}{2^n} \sum_{(x,y,z)\in R,D} \| \sum_{r\neq 0^n} (-1)^{y\cdot r} \alpha_{x,y,z,D,r} \|^2$$

We will think of p' as the norm squared of the vector u whose entries are $\sum_{r\neq 0^n}(-1)^{y\cdot r}\alpha_{x,y,z,D,r}/\sqrt{2^n}$ as $(x,y,z)\in R, D$ vary. On the other hand, imagine performing an additional query to determine if $H(x) = y$. The query first applies $\mathsf{StdDecomp}$, arriving at the state

$$\sum_{r,(x,y,z)\in R,D} \alpha_{x,y,z,D,r} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y'\cdot r} |x, y, z, D \cup (x, y')\rangle$$

Then, it applies the query CStO', writing the output into a new register:

$$\sum_{r,(x,y,z)\in R,D} \alpha_{x,y,z,D,r} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y'\cdot r} |x, y, y', z, D \cup (x, y')\rangle$$

Next, $\mathsf{StdDecomp}$ is applied again, and then y, y' are measured. Since $\mathsf{StdDecomp}$ does not affect the y, y' registers, it can be ignored. Therefore, the probability p that $y = y'$ is:

$$p = \frac{1}{2^n} \sum_{(x,y,z)\in R,D} \| \sum_{r} (-1)^{y\cdot r} \alpha_{x,y,z,D,r} \|^2$$

We will think of p as the norm squared of the vector v whose entries are $\sum_r (-1)^{y\cdot r}\alpha_{x,y,z,D,r}/\sqrt{2^n}$. Let $w = v - u$, which is just the vector of entries $\alpha_{x,y,z,D,0}/\sqrt{2^n}$. Therefore, by normalization, we know that w has norm at most $1/\sqrt{2^n}$. By the triangle inequality, $\sqrt{p} = |v| \leq |u| + |w| \leq \sqrt{p'} + \sqrt{1/2^n}$, as desired.

To generalize for $k \geq 1$, we modify the above proof as follows. x, y, r will be lists of k elements. We obtain the following identities:

$$1 = \sum_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}} \|\alpha_{\mathbf{x},\mathbf{y},\mathbf{z},D,\mathbf{r}}\|^2$$

$$p' = \frac{1}{2^{kn}} \sum_{(\mathbf{x},\mathbf{y},z)\in R,D} \| \sum_{\mathbf{r}\in Z} (-1)^{\mathbf{y}\cdot\mathbf{r}} \alpha_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}} \|^2$$

$$p = \frac{1}{2^{kn}} \sum_{(\mathbf{x},\mathbf{y},z)\in R,D} \| \sum_{\mathbf{r}} (-1)^{\mathbf{y}\cdot\mathbf{r}} \alpha_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}} \|^2$$

Where Z is the set of $\mathbf{r}$ that are non-zero in every position. We will think of p' as the norm squared of the vector u whose entries are $\sum_{\mathbf{r}\in Z}(-1)^{\mathbf{y}\cdot\mathbf{r}}\alpha_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}}/\sqrt{2^{kn}}$, and p as the norm squared of the vector v whose entries are $\sum_{\mathbf{r}}(-1)^{\mathbf{y}\cdot\mathbf{r}}\alpha_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}}/\sqrt{2^{kn}}$.

Let $\overline{Z}$ be the set of $\mathbf{r}$ that are zero in at least 1 position. Let $\overline{Z}_i$ be the set of $\mathbf{r}$ such that $r_i = 0$, but $r_j \neq 0$ for $j < i$. Then $\overline{Z}_i$ partition the space $\overline{Z}$. Also note that $\overline{Z}_i$ contains at most $2^{(k-1)n}$ values; as such $\overline{Z}$ contains at most $k2^{(k-1)n}$ terms.

Next, note that $w = v - u$ is the vector with entries $\frac{1}{\sqrt{2^{kn}}}\sum_{\mathbf{r}\in\overline{Z}_i}(-1)^{\mathbf{y}\cdot\mathbf{r}}\alpha_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}}$. We now claim that w has norm at most $\sqrt{k/2^n}$. Toward that end, let $\beta_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}} = (-1)^{\mathbf{y}\cdot\mathbf{r}}\alpha_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}}$. Then $\sum_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}}\|\beta_{\mathbf{x},\mathbf{y},\mathbf{z},D,\mathbf{r}}\|^2 = 1$. On the other hand,

$$\begin{aligned}
|w| &= \frac{1}{2^{kn}}\sum_{\mathbf{x},\mathbf{y},z,D}\left\|\sum_{\mathbf{r}\in\overline{Z}}\beta_{\mathbf{x},\mathbf{y},\mathbf{z},D,\mathbf{r}}\right\|^2 \\
&\leq \frac{1}{2^{kn}}\sum_{\mathbf{x},\mathbf{y},z,D}|\overline{Z}|\sum_{\mathbf{r}\in\overline{Z}}\|\beta_{\mathbf{x},\mathbf{y},\mathbf{z},D,\mathbf{r}}\|^2 \\
&\leq \frac{k}{2^n}\sum_{\mathbf{x},\mathbf{y},z,D,\mathbf{r}\in\overline{Z}}\|\beta_{\mathbf{x},\mathbf{y},\mathbf{z},D,\mathbf{r}}\|^2 \leq 1
\end{aligned}$$

By the triangle inequality, $\sqrt{p} = |v| \leq |u| + |w| \leq \sqrt{p'} + \sqrt{k/2^n}$. □

B.4 Complete Description of our Simulator

Here, we complete the description of our simulator for the simple domain extender from Section 5.

Sim will keep a (superposition over) database D_a, which represents the simulation of the random oracle h_a that it will update according to the CStO update procedure. D_a is originally just the empty database. It will also have a private random oracle h_b. For concreteness, h_b will be implemented using another instance of CStO, but we will often think of h_b as being a uniformly random function.

On h_1 queries, *Sim* makes a query to h_a, performing the appropriate CStO update procedure to D_a.

On h_2 queries, *Sim* performs a unitary operation which maps the basis states $|x, y\rangle \otimes |D_a\rangle$ to:

$$\begin{aligned}
&|x, y \oplus h_b(x)\rangle \otimes |D_a\rangle \text{ if } \mathsf{FindInput}(x, D_a) = (0, 0^{3n}) \\
&|x, y \oplus H(w)\rangle \otimes |D_a\rangle \text{ if } \mathsf{FindInput}(x, D_a) = (1, w)
\end{aligned}$$

This unitary is implemented by the following procedure:

– Initialize a new $3n + 1$ qubit register to 0. Call these registers b, w.
– Evaluate $\mathsf{FindInput}$ in superposition, XORing the output into the newly created b, w registers.
– Initialize a new n qubit register to 0, and apply Hadamard so that the new register has the state $\frac{1}{\sqrt{2^n}}\sum_z |z\rangle$.

– Apply the following conditional swap operation. The controlling bit is b, and the two sets of registers to swap if $b = 1$ are the adversary's query response registers y and the newly created z registers.
– Make an H query on the w, z registers (here z is either the newly created register if $b = 0$, or the adversary's y register if $b = 1$).
– Make an h_b query on the x, y registers (where y is the adversary's y register if $b = 0$ or the newly created z register if $b = 1$).
– Apply the conditional swap operation a second time.
– Apply Hadamard to the z registers, which are guaranteed to be in uniform superposition, so the result is $|0\rangle$. Discard these registers.
– Uncompute $\mathsf{FindInput}$ by evaluating a second time and XORing into the b, w registers. At this point the registers are guaranteed to contain 0^{3n+1}, so they can be discarded.

B.5 Proof of Lemma 7

Lemma 5. *Consider a quantum system over x, D, x', z. The following two unitaries $O(1/\sqrt{2^n})$-almost commute:*

– $\mathsf{StdDecomp}$*, acting on the x, D registers.*
– $\mathsf{FindInput}$*, taking as input the D, x' registers and XORing the output into z.*

Proof. Let $\Delta = \mathsf{StdDecomp} \circ \mathsf{FindInput} - \mathsf{FindInput} \circ \mathsf{StdDecomp}$. Notice that since $\mathsf{StdDecomp}$ is an involution, $\mathsf{StdDecomp} \circ \Delta = -\Delta \circ \mathsf{StdDecomp}$. Write $x' = (y, x_2)$. We now examine several cases.

1. Let D be such that $\mathsf{FindInput}(D, (y, x_2)) = (1, (x_1, x_2))$, meaning that $D(x_1) = y$, and x_1 is the lowest such input. Suppose $x > x_1$ and $D(x) = \bot$.
Consider the action of Δ on $|x, D, x', z\rangle$ or $|x, D \cup (x, y'), x', z\rangle$. Notice that after applying $\mathsf{StdDecomp}$, in either case the database superposition will still have support on databases where $D(x_1) = y$ and x_1 is the lowest such input. Therefore, $\mathsf{StdDecomp}$ does not effect the output of $\mathsf{FindInput}$. Therefore, for these inputs $\Delta|x, D, x', z\rangle = 0$.
Let P_1 be the projection onto x, D, x', z such that $\mathsf{FindInput}$ finds a completion (x_1, x_2) such that $x_1 < x$. Then for any state $|\psi\rangle$, $\Delta P_1|\psi\rangle = 0$.
2. Let D be such that $\mathsf{FindInput}(D, (y, x_2)) = (1, (x_1, x_2))$ and $D(x) = \bot$, but now consider $x < x_1$ (note that since $D(x) = \bot \neq y = D(x_1)$, x must be different from x_1).
Now we have that $\mathsf{StdDecomp} \circ \mathsf{FindInput}|x, D, x', z\rangle = \mathsf{StdDecomp}|x, D, x', z \oplus (1, x_1, x_2)\rangle = \frac{1}{\sqrt{2^n}} \sum_{y'} |x, D \cup (x, y'), x', z \oplus (1, x_1, x_2)\rangle$. On the other hand, $\mathsf{FindInput} \circ \mathsf{StdDecomp}|x, D, x', z\rangle = \mathsf{FindInput} \frac{1}{\sqrt{2^n}} \sum_{y'} |x, D \cup (x, y'), x', z\rangle = \frac{1}{\sqrt{2^n}} \Big(|x, D \cup (x, y'), x', z \oplus (1, x, x_2) + \sum_{y' \neq y} |x, D \cup (x, y'), x', z \oplus (1, x_1, x_2)\rangle \Big)$.
Therefore, $\Delta|x, D, x', z\rangle = \frac{1}{\sqrt{2^n}} |x, D \cup (x, y), x'\rangle (|z \oplus (1, x_1, x_2)\rangle - |z \oplus (1, x, x_2)\rangle$.
Let P_2 be the projection onto x, D, x', z such that $D(x) = \bot$ and $\mathsf{FindInput}$ finds a completion (x_1, x_2) such that $x_1 > x$. Let M_1 be the unitary mapping

$|x, D, (y, x_2), z\rangle \mapsto |x, D \cup (x, y), (y, x_2), z \oplus \mathsf{FindInput}(D, (y, x_2))\rangle$. Let M_2 be the unitary mapping $|x, D, (y, x_2), z\rangle \mapsto |x, D \cup (x, y), (y, x_2), z \oplus (1, x, x_2)\rangle$. Then for any state $|\psi\rangle$, $\mathit{\Delta} P_2|\psi\rangle = \frac{1}{\sqrt{2^n}}(M_1 - M_2)P_2|\psi\rangle$. Therefore, $\|\mathit{\Delta} P_2|\psi\rangle\| \leq \frac{2}{\sqrt{2^n}}$.

3. Let D, x, x_1 be as in Case 2 above. Now we use the fact that $\mathsf{StdDecomp} \circ \mathit{\Delta} = -\mathit{\Delta} \circ \mathsf{StdDecomp}$ with the previous derivation to conclude that $\mathit{\Delta} \frac{1}{\sqrt{2^n}} \sum_{y'} |x, D \cup (x, y'), x', z\rangle = -\mathsf{StdDecomp} \frac{1}{\sqrt{2^n}} |x, D \cup (x, y), x'\rangle(|z \oplus (1, x_1, x_2)\rangle - |z \oplus (1, x, x_2)\rangle)$
Let P_3 be the projection onto states of the form $\sum_{x,D,x',z,y'} \alpha_{x,D,x',z}|x, D \cup (x, y'), x', z\rangle$ where the support is over D such that $D(x) = \perp$, and $\mathsf{FindInput}(D, x')$ finds a completion (x_1, x_2) such that $x_1 > x$. Then by a similar argument to Case 2, $\|\mathit{\Delta} P_3|\psi\rangle\| \leq \frac{2}{\sqrt{2^n}}$.
4. Let D, x, x_1 be as in Case 2 above. Let $r \neq 0$

$$\begin{aligned}
&\mathsf{StdDecomp} \circ \mathsf{FindInput} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle \\
&= \mathsf{StdDecomp} \frac{1}{\sqrt{2^n}} \Bigg((-1)^{y \cdot r} |x, D \cup (x, y), x', z \oplus (1, x, x_2)\rangle \\
&\qquad + \sum_{y' \neq y} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z \oplus (1, x_1, x_2)\rangle \Bigg) \\
&= \mathsf{StdDecomp} \frac{1}{\sqrt{2^n}} \Bigg(\sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z \oplus (1, x_1, x_2)\rangle \\
&\qquad + (-1)^{y \cdot r} |x, D \cup (x, y), x', z \oplus (1, x, x_2)\rangle - (-1)^{y \cdot r} |x, D \cup (x, y), x', z \oplus (1, x_1, x_2)\rangle \Bigg) \\
&= \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z \oplus (1, x_1, x_2)\rangle \\
&\qquad + \mathsf{StdDecomp} \frac{(-1)^{y \cdot r}}{\sqrt{2^n}} |x, D \cup (x, y), x'\rangle \left(|z \oplus (1, x, x_2)\rangle - |z \oplus (1, x_1, x_2)\rangle \right)
\end{aligned}$$

Meanwhile

$$\begin{aligned}
&\mathsf{FindInput} \circ \mathsf{StdDecomp} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle \\
&= \mathsf{FindInput} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle \\
&= \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z \oplus (1, x_1, x_2)\rangle \\
&\qquad + \frac{(-1)^{y \cdot r}}{\sqrt{2^n}} |x, D \cup (x, y), x'\rangle \left(|z \oplus (1, x, x_2)\rangle - |z \oplus (1, x_1, x_2)\rangle \right)
\end{aligned}$$

In other words, $\Delta \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle = (\mathbf{I} - \mathsf{StdDecomp}) \frac{(-1)^{y \cdot r}}{\sqrt{2^n}} |x, D \cup (x, y), x'\rangle \left(|z \oplus (1, x, x_2)\rangle - |z \oplus (1, x_1, x_2)\rangle\right)$.
Next, we observe that

$$\begin{aligned}
\mathsf{StdDecomp}_x |D \cup (x, y')\rangle &= \mathsf{StdDecomp}_x \frac{1}{2^n} \sum_s (-1)^{s \cdot y'} \sum_{y''} (-1)^{s \cdot y''} |D \cup (x, y'')\rangle \\
&= \frac{1}{2^n} \sum_{s \neq 0} (-1)^{s \cdot y'} \sum_{y''} (-1)^{s \cdot y''} |D \cup (x, y'')\rangle + \frac{1}{\sqrt{2^n}} |D\rangle \\
&= |D \cup (x, y')\rangle + \frac{1}{\sqrt{2^n}} \left(|D\rangle - \frac{1}{\sqrt{2^n}} \sum_{y''} |D \cup (x, y'')\rangle \right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
&\Delta \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle \\
&= \frac{(-1)^{y \cdot r}}{2^n} \left(|x, D, x'\rangle - \frac{1}{\sqrt{2^n}} \sum_{y''} |x, D \cup (x, y''), x'\rangle \right) \left(|z \oplus (1, x, x_2)\rangle - |z \oplus (1, x_1, x_2)\rangle\right)
\end{aligned}$$

Let $P_{4,r}$ be the projection onto states of the form $\sum_{x,D,x',z,y'} \alpha_{x,D,x',z} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle$ where the support is over D such that $D(x) = \perp$, and $\mathsf{FindInput}(D, x')$ finds a completion (x_1, x_2) such that $x_1 > x$. Let $P_4 = \sum_{r \neq 0} P_{4,r}$. Notice that the $P_{4,r}$'s have orthogonal support.
We can define 4 unitaries:
- $M_{3,r} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle = |x, D, x', z \oplus (1, x, x_2)\rangle$
- $M_{4,r} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle = |x, D, x', z \oplus (1, x_1, x_2)\rangle$
- $M_{5,r} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle = \frac{1}{\sqrt{2^n}} \sum_{y''} |x, D \cup (x, y''), x', z \oplus (1, x, x_2)\rangle$
- $M_{6,r} \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle = \frac{1}{\sqrt{2^n}} \sum_{y''} |x, D \cup (x, y''), x', z \oplus (1, x_1, x_2)\rangle$

Then we have that $\Delta \circ P_{4,r} |\psi\rangle = \frac{1}{2^n} (M_{3,r} - M_{4,r} - M_{5,r} + M_{6,r}) P_{4,r} |\psi\rangle$. This means that $\|\Delta \circ P_{4,r} |\psi\rangle\| \leq \frac{4}{2^n} \|P_{4,r} |\psi\rangle\|$.
Now, this means that

$$\begin{aligned}
\|\Delta \circ P_4 |\psi\rangle\| &\leq \frac{4}{2^n} \sum_{r \neq 0} \|P_{4,r} |\psi\rangle\| \\
&\leq \frac{4}{\sqrt{2^n}} \|\sum_{r \neq 0} P_{4,r} |\psi\rangle\| = \frac{4}{\sqrt{2^n}} \|P_4 |\psi\rangle\| \leq \frac{4}{\sqrt{2^n}}
\end{aligned}$$

5. Let D be such that $\mathsf{FindInput}(D, x') = (0, 0, 0)$. Then by a similar calculation to Case 4, we have that $\Delta \frac{1}{\sqrt{2^n}} \sum_{y'} (-1)^{y' \cdot r} |x, D \cup (x, y'), x', z\rangle = (\mathbf{I} - \mathsf{StdDecomp}) \frac{1}{\sqrt{2^n}} \left(|x, D \cup (x, y), x', z \oplus (1, x, x_2)\rangle - |x, D \cup (x, y), x', z \oplus (0, 0, 0)\rangle\right)$

Let $P_{5,r}$ be the projection onto states of the form $\sum_{x,D,x',z,y'} \alpha_{x,D,x',z}(-1)^{y'\cdot r}|x, D\cup (x,y'), x', z\rangle$ where the support is over D such that $D(x) = \bot$, and $\mathsf{FindInput}(D, x') = (0,0,0)$. Let $P_5 = \sum_{r\neq 0} P_{5,r}$. Notice that the $P_{5,r}$'s have orthogonal support.

By an analogous calculation to Case 4, we have that $\|\Delta\circ P_4|\psi\rangle\| \leq \frac{4}{\sqrt{2^n}}\|P_5|\psi\rangle\|$

Now, we observe that $P_1+P_2+P_3+P_4+P_5 = \mathbf{I}$. Therefore, $\|\Delta|\psi\rangle\| \leq \sum_{i=1}^{5} \|\Delta\circ P_i|\psi\rangle\| \leq \frac{12}{\sqrt{2^n}}$. By Lemma 1, this means that for any pure state, the results of applying $\mathsf{FindInput}\circ\mathsf{StdDecomp}$ and $\mathsf{StdDecomp}\circ\mathsf{FindInput}$ are at most $\frac{24}{\sqrt{2^n}}$ in trace distance. Then the same holds true for any mixed state since any mixed state is in the convex hull of pure states. This means that $\mathsf{FindInput}$ and $\mathsf{StdDecomp}$ $\frac{24}{\sqrt{2^n}}$-almost commute. This completes the proof. □

C Other Oracle Variations

Here, we describe some more oracle variations. While we do not use them in this work, they give a different perspective on our compressed oracles.

Fourier Oracle. This oracle operates on x, z, D registers, where x is m qubits, z is n qubits, and D is $n2^m$ qubits where D is interpreted as the truth table of a function from m bits to n bits. On basis states $|x,z\rangle \otimes |D\rangle$, it performs the map: $|x,z\rangle\otimes|D\rangle \mapsto |x,z\rangle\otimes|D\oplus P_{x,z}\rangle$. Here, $P_{x,z}$ is the point function that outputs z on input x, and 0 everywhere else. $D\oplus P_{x,z}$ is the function $(D\oplus P_{x,z})(x') = D(x')\oplus P_{x,z}(x')$.

For initialization, we will have that the $|D\rangle$ registers are initialized to the all-zeros function $|0\rangle$. We will call this oracle the Fourier Phase Oracle, $\mathsf{FourierPhsO}$.

Lemma 23. PhO *and* $\mathsf{FourierPhsO}$ *are perfectly indistinguishable. That is, for any adversary B making oracle queries,* $\Pr[B^{\mathsf{PhO}}() = 1] = \Pr[B^{\mathsf{FourierPhsO}}() = 1]$

Proof. Basically, $\mathsf{FourierPhsO}$ is identical to PhO, except that between queries we encode the oracle registers by applying Hadamard to the oracle's state. More precisely, consider running B^{PhO}, but before the first query and between each query, we will perform Hadamard twice on the oracle's registers. Since Hadamard is an involution, the two applications cancel out.

Now, consider the states between each pair of Hadamards, which we will label $|x,z\rangle\otimes|D\rangle$. The first such state (that is, after the very first Hadamard), the oracle will have been mapped to the all-zeros function $|0\rangle$. For subsequent "between" states, we map from one to the next by applying Hadamard, then PhO, then Hadamard again. On a basis state $|x, z\rangle \otimes |D\rangle$, the result is:

$$\begin{aligned}
|x, z\rangle \otimes |D\rangle &\mapsto \frac{1}{\sqrt{2^{n2^m}}} \sum_E (-1)^{D \cdot E} |x, z\rangle \otimes |E\rangle \text{ (First Hadamard)} \\
&\mapsto \frac{1}{\sqrt{2^{n2^m}}} \sum_E (-1)^{D \cdot E + z \cdot E(x)} |x, z\rangle \otimes |E\rangle \text{ (Apply Phase Oracle)} \\
&= \frac{1}{\sqrt{2^{n2^m}}} \sum_E (-1)^{(D \oplus P_{x,z}) \cdot E} |x, z\rangle \otimes |E\rangle \\
&\mapsto |x, z\rangle \otimes |D \oplus P_{x,z}\rangle \text{ (Second Hadamard)}
\end{aligned}$$

Hence, by considering these intermediate states, we obtain $\mathsf{FourierPhsO}$. □

Whether the oracle is implemented in the computational of Fourier domains is orthogonal to whether the interface provided is the computational or phase domains. More precisely, we can consider a Fourier *standard* oracle $\mathsf{FourierStO}$, which is obtained from $\mathsf{FourierPhsO}$ by applying $H^{\otimes n}$ to the response registers before and after each query. The following Lemma follows immediately:

Lemma 24. StO *and* $\mathsf{FourierStO}$ *are perfectly indistinguishable. That is, for any adversary A making oracle queries,* $\Pr[A^{\mathsf{StO}}() = 1] = \Pr[A^{\mathsf{FourierStO}}() = 1]$

Compressed Fourier Oracle. Consider the Fourier Phase oracle $\mathsf{FourierPhsO}$ above. D starts off as the all-zeros function O. On each query, D will be XORed with a point function. Therefore, after q queries, D will only have support on functions that are the sum of q point functions — in particular, it will be zero in all but q locations. Therefore, we can actually compress D into a list of at most q pairs (x, z) with distinct x such that $z \neq 0$. For concreteness, we will represent D as a list $(x_1, z_1), (x_2, z_2), \dots,$ where $x_1 < x_2 < \dots$ and $z_i \neq 0$ for all i. For a compressed D, a pair $(x, z) \in D$ corresponds to $D(x) = z$, and if there is no pair whose first term is x, this corresponds to $D(x) = 0$ in the uncompressed D.

Let $\emptyset$ denote the empty list. We will say that $D(x) = y$ if there is a pair (x, y) in the list D. We will say that $D(x) = \bot$ if there is no such pair. Since all pairs have unique x, $D(x)$ is a function. For a pair (x, y) where $D(x) = \bot$, we will define $D \cup (x, y)$ to be the list obtained by inserting (x, y) into D. For $(x, y) \in D$, we will define $D \setminus (x, y)$ to be the list obtained by removing (x, y) from D.

Finally, we will define $D \oplus (x, z)$, which corresponds to XORing the function $P_{x,z}$ to the un-compressed function. It is straightforward to see that the following definition carries out the desired operation:

$$D \oplus (x, z) = \begin{cases} D & \text{if } z = 0 \\ D \cup (x, z) & \text{if } z \neq 0 \text{ and } D(x) = \bot \\ D \setminus (x, z) & \text{if } z \neq 0 \text{ and } D(x) = z \\ (D \setminus (x, z')) \cup (x, z \oplus z') & \text{if } z \neq 0 \text{ and } D(x) = z' \notin \{z, \bot\} \end{cases}$$

Notice that the map $(x, z), D \mapsto (x, z), D \oplus (x, z)$ is an involution which can be computed by the following reversible procedure:

1. First, if $D(x) = y \notin \{\perp, 0\}$, then do nothing. If $D(x) = \perp$, then perform the map $D \mapsto D \cup (x, 0)$. Note that this map is not reversible in general, but since D was guaranteed to not contain any pairs whose second coordinate is zero, D represents a reversible operation on valid initial D. More precisely, we can define the map on illegal lists to be $D \cup (x, 0) \mapsto D$ for D such that $D(x) = \perp$. Then the overall map is an involution.
   This map can be seen as a sort of local decompression for D. If $D(x) = \perp$, it decompresses to D' such that $D'(x) = 0$ and vice versa. If $D(x) \notin \{0, \perp\}$, the D is incompressible at that point so it is left unchanged. We will call this procedure $\mathsf{FourDecomp}$, for decompressing our Fourier oracle. We will use $\mathsf{FourDecomp}$ to be the two-input reversible function acting on (x, D), and $\mathsf{FourDecomp}_x$ to be the one-input conditional function acting on D based on the value x.
2. Now, we are guaranteed that D contains a pair (x, z') for some z' (which may be 0). Perform the map which replaces this pair with $(x, z \oplus z')$.
3. Finally, we reverse the first step. Notice that $D(x) = z \oplus z'$, which may be 0, but is never $\perp$. Therefore, if $D(x) = 0$, we delete the pair $(x, 0)$ from D. In general, this map is not reversible, but it is guaranteed to be reversible on the set of possible D that are the results of step (2). Namely, by specifying the map on illegal D (namely those for which $D(x) = \perp$) as $D \mapsto D \cup (x, 0)$, we make the map an involution. This step is actually identical to step (1), but has the opposite effect on legal states: it intuitively locally re-compressed the portion of D corresponding to input x.

Instead of decompressing, applying the Fourier oracle, and decompressing, we can instead describe how the map behaves directly on the compressed encoding. This gives us the compressed Fourier phase oracle $\mathsf{CFourierPhsO}$. It starts with an empty database $D = \emptyset$, the result of compressing the all-zeros function. Moreover, on each query it performs the map: $|x, z\rangle \otimes |D\rangle \mapsto |x, z\rangle \otimes |D \oplus (x, z)\rangle$.

We can imagine implementing the operation above in the three steps corresponding to the three steps to compute $D \oplus (x, z)$ above. Notice that the first and last step involve a computational basis test on the registers containing $D(x)$ to test if they contain 0.

Note that as described above, the compressed Fourier oracle implements a phase query to the adversary. We can also imagine it implementing a standard query, obtaining the compressed Fourier Standard oracle $\mathsf{CFourierStO}$ by applying Hadamard to the adversary's response registers before and after each query. The following lemma follows immediately form the above discussion.

Lemma 25. $\mathsf{CFourierPhsO}$ *and* $\mathsf{FourierPhsO}$ *are perfectly indistinguishable.* $\mathsf{CFourierStO}$ *and* $\mathsf{FourierStO}$ *are perfectly indistinguishable. That is, for any adversary A, we have* $\Pr[A^{\mathsf{CFourierStO}}() = 1] = \Pr[A^{\mathsf{FourierStO}}() = 1]$*, and for any adversary B, we have* $\Pr[B^{\mathsf{CFourierPhsO}}() = 1] = \Pr[B^{\mathsf{FourierPhsO}}() = 1]$.

Another derivation of our compressed standard and phase oracles is the following: start with a compressed Fourier oracle (standard or phase), and then encode the oracle's state between queries by applying the Hadamard transformation to all the y registers in the database. The result is exactly the same as the compressed oracle given in Section 3.

D Quatum Indifferentiability of Merkle-Damgård

In this section, we use our compressed oracle technique to prove the indifferentiability of the full Merkle-Damgård construction when using a pre-fix free encoding.

D.1 Background

Pre-fix free encoding. A prefix-free code over $\{0,1\}^*$ is a set S such that, for all $x \in S, x \neq y$, x is not a prefix of y.

Merkle-Damgård. We briefly recall the Merkle-Damgård construction. Let $h : \{0,1\}^{2n} \to \{0,1\}^n$ be a compression function. Let IV be an initial value. We can consider IV to be 0. Let S be a prefix-free code over $(\{0,1\}^n)^*$. Given an input $x \in S$, define $\mathsf{MD}_h(w)$ as follows. First, write w as $(w_1, \dots, w_\ell)$, where each $w_i \in \{0,1\}^n$. Then:

- Let $z_0 = IV$.
- For each $i = 1, \dots, \ell$, let $z_i = h(z_{i-1}, w_i)$.
- Output z_ℓ.

D.2 Our Simulator

We now prove the quantum indifferentiability of Merkle-Damgård:

In order to describe our simulator, we will need to modify some of the definitions/algorithms from Section 5.

Definition 6. *Let D be a database of (x, y) pairs. Fix an $IV \in \{0,1\}^n$. A "completion" for an input $x \in \{0,1\}^{2n}$ is a list of tuples $(x_i, z_i) \in D$ for $i = 1, \dots, \ell - 1$ such that, if we write $x_i = (z'_{i-1}, w_i) \in \{0,1\}^n \times \{0,1\}^{n-m}$ and $x = (z'_{\ell-1}, w_\ell)$,*

- *$z'_i = z_i$ for $i > 0$ and $z'_0 = IV$*
- *$(w_1, \dots, w_\ell)$ is in the prefix-free code S.*

We will say that D is *good* if (1) it contains no collision, and (2) $y_i \neq IV$ for all i. The following lemma is implicit in the work of [CDMP05]:

Lemma 26. *Let D be a good database. Then for any input $x \in \{0,1\}^n$, D contains at most a single completion. Moreover, if x's completion is $\{(x_i, z_i)\}_i$ for $i = 1, \dots, \ell - 1$, then none of the x_i have a completion in D.*

Also implicit in [CDMP05] is an algorithm $\mathsf{FindCompletion}$ which, on input x, D, will return a bit $b \in \{0,1\}$ indicating if it found a completion, and a string $w = (w_1, \ldots, w_\ell)$ representing the completion. If no completion is found, $w = \emptyset$.

Our simulator is defined as follows. The simulator Sim implements h as the compressed standard oracle CStO, but will make occasional exceptions in order to make sure the oracle is "consistent with" H. Sim maintains a superposition over databases D of (x, y) pairs. D is initially empty. On each query, Sim does the following:

– Run $\mathsf{FindCompletion}$ in superposition on the adversary's query registers and the stored database D, writing the output to new registers initialized to 0. That is, append $|0\rangle$ to the state, and apply the unitary

$$|x, y\rangle \otimes |D\rangle \otimes |r\rangle \mapsto |x, y\rangle \otimes |D\rangle \otimes |r \oplus \mathsf{FindCompletion}(x, D)\rangle$$

– Interpret the r registers as a pair (b, w) where b represents whether or not a completion was found, and w represents the completion if found. Perform the unitary which does the following:
  - Apply CStO, conditioned on $b = 0$, to the x, y, D registers. In other words, $|x, y\rangle \otimes |D\rangle \otimes |0, 0\rangle \mapsto (\mathsf{CStO}|x, y\rangle \otimes |D\rangle) \otimes |0, 0\rangle$.
  - Make an H query on w conditioned on $b = 1$, using y as the response register. In other words, $|x, y\rangle \otimes |D\rangle \otimes |1, w\rangle \mapsto |x, y \oplus H(w)\rangle \otimes |D\rangle \otimes |1, w\rangle$.

  The conditional queries can be implemented in a straightforward fashion analogous to the implementation in Section B.4.

The proof of indifferentiability for our simulator follows the exact same proof structure as the proof in Section 5, adapted for the setting here. In particular, we first prove that $\mathsf{FindCompletion}$ almost commutes with $\mathsf{StdDecomp}_x$ (Section D.3). Then we use this near-commutativity to prove the indistinguishability (Section D.4) and consistency (Section D.5) of Sim. Together, this implies that Sim is indifferentiable.

D.3 The Almost Commutativity of $\mathsf{StdDecomp}_x$ and $\mathsf{FindCompletion}$

Lemma 27. *Consider a quantum system over x, D, x', z where $|D| \leq q$. The following two unitaries $O(q/\sqrt{2^n})$-almost commute:*

– $\mathsf{StdDecomp}$ *applied to the x, D registers.*
– $\mathsf{FindCompletion}$*, applied to D, x', with the output XORed into z.*

Proof. $\mathsf{FindCompletion}$ can be implemented using q applications of $\mathsf{FindInput}$, one for each potential member of the completion. Since each of the $\mathsf{FindInput}$'s $O(1/\sqrt{2^n})$-almost commute with $\mathsf{StdDecomp}$, the q applications will $O(q/\sqrt{2^n})$-almost commute. □

D.4 Indistinguishability

Lemma 28. Sim *is indistinguishable. In particular, for any distinguisher $\mathcal{D}$ making at most q queries to h,*

$$|\Pr[\mathcal{D}^h() = 1] - \Pr[\mathcal{D}^{\mathsf{Sim}^H}() = 1]| < O(q^2/\sqrt{2^n})$$

Proof. The proof is analagous to the proof of Lemma 8, adapted for the current simulator. We consider a distinguisher D, and prove security through a sequence of hybrids.

Hybrid 0. This is the real world, where h is a random oracle. Let p_0 be the probability D outputs 1 in this case.

Hybrid 1. This is still the real world, but we add an abort condition. Namely, after any query to h, we measure if the database h is good; if not, we immediately abort and stop the simulation. Let p_1 be the probability $\mathcal{D}$ outputs 1 in Hybrid 1. The proof of the following lemma is essentially identical to the proof of Lemma 9:

Lemma 29. $|p_1 - p_0| \leq O(\sqrt{q^3/2^n})$

Next, we define a classical encoding procedure Encode for databases D. Intuitively, Encode will scan the values (x, y) in D, seeing if any of the x values correspond to a completion in D. If so, such a completion will have an associated input w. Encode will reasonably guess that such a completion corresponds to an evaluation of $H(w) = \mathsf{MD}_h(w)$. Therefore, Encode will remove the value (x, y) in D, and add the pair (w, z) to a new database E, intuitively representing the oracle H. In more detail, Encode does the following:

- For each pair $P = (x, y) \in D$, run $\mathsf{FindCompletion}(x, D \setminus P) = (b, w)$. If $b = 1$, re-label the pair to (w, y)
- Remove all re-labeled pairs D (which are easily identifiable since the input will be larger) and place them in a new database E.

We define the following Decode procedure, which operates on pairs D, E:

- Merge the databases D, E
- For each pair (w, y) that was previously in E, write $w = (w_1, \ldots, w_\ell)$.
  - Let $z_0 = IV$ and evaluate $z_i = D(z_{i-1}, w_i)$ for $i = 1, \ldots, \ell - 1$.
  - Re-label (w, y) to $((z_{\ell-1}, w_\ell), y)$.
  - If $z_i = \perp$ for any $i \leq \ell - 1$, output $\perp$ and abort. If $D(z_{\ell-1}, w_\ell) \neq \perp$, output $\perp$ and abort.

Note that, for good databases, $\mathsf{Encode}, \mathsf{Decode}$ are independent of the order elements are processed. It follows immediately from the descriptions above that on good databases D, $\mathsf{Decode}(\mathsf{Encode}(D)) = D$. Therefore, Encode can be implemented in superposition, giving the unitary that maps $|D\rangle$ to $|\mathsf{Encode}(D)\rangle$. Also note that $\mathsf{Encode}(\emptyset) = (\emptyset, \emptyset)$.

Hybrid 2. In this hybrid, there are two databases D, E, initialized to $|\emptyset, \emptyset\rangle$. Each query is answered in the following way:

– Apply Decode to the D, E registers. Measure if Decode gives $\perp$, in which case abort. Otherwise, there are now just one database register D.
– Answer an h query by applying the CStO update procedure to D
– Apply Encode to D.
– Apply the check for the goodness of D.

Let p_1 be the probability D outputs 1 in Hybrid 1. The proof of the following lemma is identical to that of Lemma 10:

Lemma 30. $p_1 = p_0$

Hybrid 3. This hybrid is the ideal world, where h queries are answered by Sim, but we still have the abort condition. In other words, instead of decoding, applying the query, and then encoding, in Hybrid 3 we act directly on the encoded state using the algorithms specified by Sim. h queries, on superpositions over x, y, D, E, can be summarized as follows:

1. Compute $(b, w) = \mathsf{FindCompletion}(x, D)$ in superposition, writing the output to new registers.
2. In superposition, apply the following conditional procedures:
3. Conditioned on $b = 0$,
   (a) Uncompress D at x.
   (b) Apply in superposition the map $|x, y, D, E, b, w\rangle \mapsto |x, y \oplus D(x), D, E, b, w\rangle$.
   (c) Re-compress D at x.
4. Conditioned on $b = 1$,
   (a) Uncompress E at w.
   (b) Apply in superposition the map $|x, y, D, E, b, w\rangle \mapsto |x, y \oplus E(w), D, E, b, w\rangle$.
   (c) Re-compress E at w.
5. Uncompute (b, w) by running $\mathsf{FindCompletion}(x, D)$ in superposition again.

Let p_3 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 31. $|p_3 - p_2| \leq O(q^3/\sqrt{2^n})$.

Proof. We start with the very last query, and gradually change the queries one-by-one from how they were answered in Hybrid 2 to Hybrid 3. Consider the action of the query on the registers $|x, y, D, E\rangle$.

First, recall that Encode/Decode are independent of the order in which inputs are processed, since the database is guaranteed to be good. Therefore, we will assume the Encode operation at the end of the query in Hybrid 2 processes x last, if it exists in the database. All other inputs are processed before x. One by one, we will move the processing of these $\neq x$ inputs to occur just after the initial decode but before the CStO update. We observe the following:

– The processing of an input $x' \neq x$ commutes with the CStO' map within CStO. Recall that $\mathsf{CStO}'|x, y, D, E\rangle = |x, y \oplus D(x), D, E\rangle$. Since processing $x' \neq x$ does not change the value of $D(x)$, we can process before or after without any affect.
– The processing of an input x' commutes with $\mathsf{StdDecomp}$, except for the application of $\mathsf{FindCompletion}$. But recall that by Lemma 27 these $O(q/\sqrt{2^n})$-almost commute.

The result is that, for each $x' \neq x$, that processing $O(q/\sqrt{2^n})$-almost commutes with the application of CStO. As such, we can move them to occur just before CStO and incur only a $O(q^2/\sqrt{2^n})$ error in trace distance. Then, each processing step during Encode will exactly cancel out a processing step during Decode.

Unfortunately, the above does not apply to the processing of x itself, since processing x may affect the value of $D(x)$ and hence does not in general commute with CStO'. Instead, at this point, our $\mathsf{Decode}, \mathsf{Encode}$ only process at most a single element corresponding to the input x. Thus, our modified Hybrid 2 procedure looks like:

1. Compute $(b, w) = \mathsf{FindCompletion}(x, D)$ in superposition, writing the output to new registers.
2. In superposition, apply the following conditional procedure, conditioned on $b = 1$: re-label any pair $(w, y) \in E$ to (x, y) and move it to D. Because the previous query finished with an Encode, we are guaranteed that (x, y) is not present in D.
3. Apply CStO:
   (a) Uncompress D at x.
   (b) Apply in superposition the map $|x, y, D, E, b, w\rangle \mapsto |x, y \oplus D(x), D, E, b, w\rangle$.
   (c) Re-compress D at x.
4. In superposition, apply the following conditional procedure, conditioned on $b = 1$: re-label any pair $(x, y) \in D$ to (w, y) and move it to E.
5. Uncompute (b, w) by running $\mathsf{FindCompletion}(x, D)$ in superposition again.

We now show that our modified Hybrid 2 is identical to Hybrid 3.

Fix an x, D and suppose D is good (recall that this means it has no collisions and no pairs whose y value is IV). There are two cases:

– $\mathsf{FindCompletion}(x, D) = (0, 0)$. Then in our modified Hybrid 2, decoding/encoding does not affect the labeling for any (x, z) pair in D. As such, Hybrid 2 will uncompress D at x, apply the map $|x, y, D, E\rangle \mapsto |x, y \oplus D(x), D, E\rangle$ and then re-compress D at x, for these x, D.
– $\mathsf{FindCompletion}(x, D) = (1, w)$. Then in our modified Hybrid 2, decoding will re-label a $(w, z) \in E$ (if present) to $(x, z) \in D$. The effect of Hybrid 1 in this case will be to uncompress E at w, apply the map $|x, y, D, E\rangle \mapsto |x, y \oplus E(x), D, E\rangle$, and then re-compress E at w.

In either case, the effects are identical in both modified Hybrid 2 and in Hybrid 3.

After q queries to h, the total error between Hybrid 1 and Hybrid 2 is at most $O(q^3/\sqrt{2^n})$. □

Hybrid 4. This is the ideal world, where we remove the abort condition from Hybrid 3. Let p_4 be the probability $\mathcal{D}$ outputs 1 in Hybrid 4. By an almost identical proof to that of Lemma 9, we have:

Lemma 32. $|p_4 - p_3| \leq O(\sqrt{q^3/2^n})$

Summing up, we have that $|p_4 - p_0| < O(q^3/\sqrt{2^n})$, proving Lemma 28. $\square$

D.5 Consistency

Lemma 33. Sim *is consistent. In particular, for any distinguisher* $\mathcal{D}$ *making at most* q *quantum queries to* h, H *where the queries to* H *have block length at most* ℓ,

$$|\Pr[\mathcal{D}^{\mathsf{Sim}^H, H}() = 1] - \Pr[\mathcal{D}^{\mathsf{Sim}^H, C^{\mathsf{Sim}^H}}() = 1]| < O(q^2\ell/\sqrt{2^n})$$

In other words, h are simulated as Sim^H, and the adversary has to distinguish between H and C^h.

Proof. Again, the proof is analogous to the proof of Lemma 13. We first work out how H queries are answered using C^h, when we simulate h using Sim^H. We work out the case for a query on a two-block input $x = (x_1, x_2)$, the other cases being handled analogously. The input registers will be labeled with $x = (x_1, x_2)$, and the output registers labeled with y.

1. First, make an h query on (IV, x_1), writing the output to some new registers initialized to $z = 0^n$. Since we are implementing h using CStO, this is accomplished using the following steps:
   (a) Un-compress D at (IV, x_1) by applying $\mathsf{StdDecomp}_{IV,x_1}$ to D.
   (b) Evaluate the map $|x_1, z, x_2, y\rangle \otimes |D\rangle \mapsto |x_1, z \oplus D(IV, x_1), x_2, y\rangle \otimes |D\rangle$, where z is the new register that was initialized to 0.
   (c) Re-compress D at (IV, x_1) by applying $\mathsf{StdDecomp}_{IV,x_1}$ again.
2. Next, make an h query on input (z, x_2) (where z where the registers created previously) with output registers y. This has the effect of mapping to:

$$|x_1, z, x_2, y \oplus h_b(x)\rangle \otimes |D\rangle \text{ if } \mathsf{FindInput}((z, x_2), D) = (0, 0)$$
$$|x_1, z, x_2, y \oplus H(w)\rangle \otimes |D\rangle \text{ if } \mathsf{FindInput}(z, x_2), D) = (1, w)$$

3. Finally, make another h query to un-compute the value of z. This is accomplished in the following steps:
   (a) Un-compress D at (IV, x_1) by applying $\mathsf{StdDecomp}_{IV,x_1}$ to D.
   (b) Evaluate the map $|x_1, z, x_2, y\rangle \otimes |D\rangle \mapsto |x_1, z \oplus D(IV, x_1), x_2, y\rangle \otimes |D\rangle$.
   (c) Re-compress D at (IV, x_1) by applying $\mathsf{StdDecomp}_{IV,x_1}$ again.
   (d) Then discard the z registers.

Let $\mathcal{D}$ be a potential distinguisher. We prove indistinguishability by introducing some hybrids:

Hybrid 0. In this hybrid, H queries are answered using C^h, as worked out above. Let p_0 be the probability $\mathcal{D}$ outputs 1.

Hybrid 1. This hybrid is identical to Hybrid 0 as outlined above, except that Steps 1c and 3a are removed, with analogous changes for inputs consisting of more blocks. Let p_1 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 34. $|p_1 - p_0| < O(q^2\ell/\sqrt{2^n})$.

Proof. Since Steps 1c and 3a are inverses of each other, Hybrid 1 is equivalent to moving Step 3a up to occur just after Step 1c. By Lemma 27, for each query to H this creates an error $O(q/\sqrt{2^n})$. In general, for an ℓ-block message, there will be ℓ such errors, totaling $O(q\ell/\sqrt{2^n})$ for each query, yielding a total error of $O(q^2\ell/\sqrt{2^n})$. □

Hybrid 2. This hybrid is identical to Hybrid 2, except that after each query we measure if the database D is good. If not, we abort and stop the simulation. Let p_2 be the probability $\mathcal{D}$ outputs 1 in this hybrid. By an almost identical proof to that of Lemma 9, we have:

Lemma 35. $|p_2 - p_1| < O(\sqrt{q^3/2^n})$

Hybrid 3. This hybrid is identical to Hybrid 2 as outlined above, except that:

– Steps 1c and 3a are removed (as in Hybrid 1)
– The operation in Step 2 is replaced with

$$|x_1, z, x_2, y\rangle \otimes |D\rangle \mapsto |x_1, z, x_2, y \oplus H(x_1, x_2)\rangle \otimes |D\rangle$$

In other words Hybrid 3 is identical to Hybrid 2, except that we change Step 2. Let p_3 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 36. $p_3 = p_2$.

Proof. In either hybrid, since we do not apply the Steps 1c and 3a, D is guaranteed to contain the pair $((IV, x_1), z)$, where z is the same as in Step 2. Therefore, in Hybrid 1, $\mathsf{FindCompletion}((z, x_2), D)$ is guaranteed to find a completion. Moreover, for good D (containing no collisions of y values equal to IV), $\mathsf{FindInput}((z, x_2), D)$ will find exactly the completion $((IV, x_1), z)$. In this case, $w = (x_1, x_2)$, and Hybrid 1 will make a query to H on (x_1, x_2). The end result is that for good D, Step 2 is identical in both Hybrids. Since D is always guaranteed to be good, the lemma follows. □

Hybrid 4. In this hybrid, H queries are made directly to H. Let p_4 be the probability $\mathcal{D}$ outputs 1 in this hybrid.

Lemma 37. $p_4 = p_3$

Proof. In Hybrid 2, what remains of Steps 1 and 3 are exact inverses of each other and moreover commute with the new Step 2 from Hybrid 2. Therefore, we can remove Steps 1 and 3 altogether without affecting how oracle queries are answered. The result is identical to Hybrid 3. □

Hybrid 5. This hybrid has H queries made directly to H, but without the abort condition. Let p_5 be the probability $\mathcal{D}$ outputs 1 in this hybrid. By an almost identical proof to that of Lemma 9, we have:

Lemma 38. $|p_5 - p_4| < O(\sqrt{q^3/2^n})$

Overall then $|p_0 - p_5| < O(q^2\ell/\sqrt{2^n})$, finishing the proof of Lemma 33. □

E Quantum Tests

Here, we formalize the notion of a quantum *test*. Such tests will determine if a particular register contains a given value or values (in an appropriate basis).

In more detail, for a set S and value x, let $\mathbb{1}(x \in S)$ be the function that outputs 1 if $x \in S$ and 0 otherwise. Let C be a 0/1 function on values z from some set. A *computational basis test*, denoted CBT_C, performs the unitary defined on the computational basis states as $|x, S, b, z\rangle \mapsto |x, S, b \oplus (C(z) \cdot \mathbb{1}(x \in S)), z\rangle$. In other words, conditioned on $C(z) = 1$, it will test if $x \in S$, XOring the result into the b registers. We will call x the test registers, S the set registers, b the output registers, and z the auxiliary registers.

We will also define a *Fourier basis test*, denoted FBT_C, which is identical to a computational test, except that it performs the Hadamard on the x registers before and after the test. The result is that it will test if $x \in S$ in the Fourier domain.

The two types of tests above, if they share the same test registers, do not commute. In particular, if we test if x is equal to y in the computational basis and equal to z in the Fourier basis, or perform the tests in the reverse order, the resulting states will not be the same. However, we will see that they very nearly commute, meaning interleaving the tests, while guaranteed to modify the state x, only does negligibly.

Definition 7. *Let U_0, U_1 be unitaries over the same quantum system. We say that U_0, U_1 ϵ-almost commute if, for any initial state ρ, the images of ρ under U_0U_1 and U_1U_0 are at most ϵ-far in trace distance.*

Lemma 39. *Consider a quantum system over n-bit strings x, subsets $S \subseteq \{0,1\}^n$ of size at most s, subsets $T \subseteq \{0,1\}^n$ of size at most t, output registers $b, c \in \{0,1\}$, and auxiliary information z (of arbitrary size). The following two unitaries $8\sqrt{st/2^n}$-almost commute:*

- CBT_C*, where x is the test register, S the set register, b the output register, (c, z) the auxiliary register, and conditional function C.*
- FBT_D*, where x is the test register, T the set register, c the output register, (b, z) the auxiliary register, and conditional function D.*

Proof. Define $|\phi_y\rangle = \frac{1}{\sqrt{2^n}} \sum_x (-1)^{y \cdot x} |x\rangle$. Let H_0 be the unitary that applies the Hadamard operation on the b registers, and H_1 the unitary that applies

Hadamard on the c registers. Let

$$P = \sum_{\substack{S,T,b,b',c,z,x:\\ x\in S, C(c,z)=1}} (-1)^{b+b'} |x,S,T,b',c,z\rangle\langle x,S,T,b,c,z|$$

$$Q = \sum_{\substack{S,T,b,c,c',z,x:\\ y\in T, D(b,z)=1}} (-1)^{c+c'} |\phi_y\rangle\langle\phi_y| \otimes |S,T,b,c,z\rangle\langle S,T,b,c',z|$$

A straightforward calculation shows that $\mathbf{I}-P$ is the computational basis test and $\mathbf{I}-Q$ is the Fourier basis test. We will first consider the case of pure state $|\psi\rangle$. Then the two resulting states are $(\mathbf{I}-Q)(\mathbf{I}-P)|\psi\rangle$ and $(\mathbf{I}-P)(\mathbf{I}-Q)|\psi\rangle$. The difference between these states is $(PQ-QP)|\psi\rangle$. Working out the product PQ, we see that

$$\begin{aligned}
PQ &= \frac{1}{\sqrt{2^n}} \sum_{S,T} L_{S,T} \otimes M_z \otimes N_z \otimes |S,T,z\rangle\langle S,T,z| \text{ where}\\
L_{S,T} &= \sum_{x\in S, y\in T} (-1)^{x\cdot y} |x\rangle\langle\phi_y|\\
M_z &= \sum_{b',b:D(b,z)=1} (-1)^{b+b'} |b'\rangle\langle b|\\
N_z &= \sum_{c,c':C(c,z)=1} (-1)^{c+c'} |c\rangle\langle c'|
\end{aligned}$$

Therefore, PQ is a block-diagonal matrix consisting of matrices $\frac{1}{\sqrt{2^n}} L_{S,T} \otimes M_z \otimes N_z$ on the diagonal. Now, the M_z (resp. N_z) matrices have spectral norm either 0,1, or 2, corresponding to the the number of solutions to $D(b,z)=1$ for $b\in\{0,1\}$ (resp. $C(c,z)=1$ for $c\in\{0,1\}$).

Next, notice that applying the QFT to the right side of $L_{S,T}$ and then deleting the all-zero columns and rows yields an $|S|\times|T|$ matrix with entries of absolute value 1. The spectral norm of such a matrix, and hence the spectral norm of $L_{S,T}$, is at most $\sqrt{|S|\times|T|}$.

Putting it all together, the spectral norm of PQ is at most the spectral norm of one of the matrices on the diagonal, which is at most $4\sqrt{st/2^n}$. This means that the norm of $(PQ-QP)|\psi\rangle$ is at most $8\sqrt{st/2^n}$. Applying Lemma 1, we see that the states τ,τ' have trace distance at most $8\sqrt{st/2^n}$.

This proves the lemma for the case of pure states. For the case of general mixed states ρ, write ρ as a convex combination of pure states, apply the lemma to each pure state, and then use to triangle inequality to bound the overall trace distance between τ,τ' as $8\sqrt{st/2^n}$.

F Quantum Security of Fujisaki-Okamoto

In this section, we use our compressed oracle technique to prove the security of the Fujisaki-Okamoto [FO99] transformation in the quantum random oracle model of Boneh et al. [BDF+11].

F.1 Background

For a random variable X, let $H_\infty(X) = \max_x(-\log \Pr[X = x])$ be the min-entropy of X.

The building blocks for the Fujisaki-Okamoto (FO) transformation are:

Symmetric key encryption. A symmetric key encryption scheme is a pair of PPT algorithms $(\mathsf{Enc}, \mathsf{Dec})$ such that:

– $\mathsf{Enc}(k, m)$ takes as input a key $k \in \{0,1\}^\lambda$ and a message m, and produces a ciphertext c
– $\mathsf{Dec}(k, c)$ takes as input a key k and ciphertext c, and produces either a message m or a special symbol $\perp$ indicating rejection.
– **Correctness:** For any key k and message m,

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

– **One-time security:** For any quantum polynomial time adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$|\Pr[\texttt{OT-Exp}_0(\lambda, \mathcal{A}) = 1] - \Pr[\texttt{OT-Exp}_1(\lambda, \mathcal{A}) = 1]| < \mathsf{negl}(\lambda)$$

where $\texttt{OT-Exp}_b(\lambda, \mathcal{A})$ is the following experiment:
- The challenger chooses a random key $k \in \{0,1\}^\lambda$
- The adversary $\mathcal{A}$, on input λ, produces two messages m_0^*, m_1^* such that $|m_0^*| = |m_1^*|$ and sends them to the challenger.
- The challenger computes $c^* \leftarrow \mathsf{Enc}(k, m_b^*)$ and returns it to the adversary.
- The adversary outputs a guess b' for b.

Public key encryption. A public key encryption scheme is a triple of PPT algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

– $\mathsf{Gen}(\lambda)$ takes as input the security parameter and produces a secret key/public key pair $(\mathsf{sk}, \mathsf{pk})$
– $\mathsf{Enc}(\mathsf{pk}, m)$ takes as input a public key pk and a message m, and produces a ciphertext c
– $\mathsf{Dec}(\mathsf{sk}, c)$ takes as input a secret key sk and ciphertext c, and produces either a message m or a special symbol $\perp$ indicating rejection.
– **Correctness:** For any message m,

$$\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$$

– **Well-spread:** There exists a super-logarithmic function p such that, for any pk produced by $\mathsf{Gen}(\lambda)$ and any message m,

$$H_\infty(\mathsf{Enc}(\mathsf{pk}, x)) \geq p(\lambda)$$

In other words, the probability of any particular ciphertext is negligibly small.

– **One-way security:** Fix a message length $n = n(\lambda)$ that is polynomial in λ. For any quantum polynomial time adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$\Pr[\texttt{OW-Exp}(\lambda, \mathcal{A}) = 1]| < \mathsf{negl}(\lambda)$$

where $\texttt{OW-Exp}(\lambda, \mathcal{A})$ is the following experiment:
  - The challenger chooses a random key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(\lambda)$, and sends pk to $\mathcal{A}$.
  - The challenger then chooses a uniformly random message m of length n, and sends $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ to $\mathcal{A}$
  - The adversary responds with a guess m' for m.
  - The challenger outputs 1 if $m = m'$ and 0 otherwise.

CCA-secure public key encryption in the quantum random oracle model. The result of of the FO transformation is a public key encryption scheme, but with the following modifications:

– **Random oracle model.** The algorithms $\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}$ all make (classical) queries to a function $H : \{0,1\}^a \to \{0,1\}^b$.

– **Security under a quantum chosen ciphertext attack in the quantum random oracle model.** This is an adaptation of the quantum CCA security definition of Boneh and Zhandry [BZ13] to the random oracle model.
For any quantum polynomial time adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that

$$|\Pr[\texttt{CCA-RO-Exp}_0(\lambda, \mathcal{A}) = 1] - \Pr[\texttt{CCA-RO-Exp}_1(\lambda, \mathcal{A}) = 1]| < \mathsf{negl}(\lambda)$$

where $\texttt{CCA-RO-Exp}_b(\lambda, \mathcal{A})$ is the following experiment:
  - The challenger chooses a random function $H : \{0,1\}^a \to \{0,1\}^b$. The challenger chooses a random key pair $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}^H(\lambda)$, and sends pk to $\mathcal{A}$.
  - The adversary is allowed to make the following queries:
    - ∗ **Quantum Random Oracle:** The adversary makes a quantum oracle query to H. For concreteness, we will assume H is implemented as the standard oracle, though it is equivalent to consider the phase oracle. $\mathcal{A}$ can make as many queries to H as it would like.
    - ∗ **Challenge query:** The adversary chooses two messages m_0^*, m_1^* such that $|m_0^*| = |m_1^*|$ and sends them to the challenger. The challenger computes $c^* = \mathsf{Enc}(\mathsf{pk}, m_b^*)$, and returns it to the adversary. For simplicity, we will restrict $\mathcal{A}$ to making only a single challenge query, though a straightforward hybrid argument will show that this is equivalent to allowing arbitrarily many challenge queries.

* **CCA queries:** The adversary makes a quantum query to the function CCA, defined as

$$CCA(c) = \begin{cases} \bot & \text{if } c \text{ was the result of a previous challenge query} \\ \mathsf{Dec}(\mathsf{sk}, c) & \text{otherwise} \end{cases}$$

The adversary can make as many CCA queries as it would like.
- Finally, the adversary produces a guess b' for b.

The FO Transformation. Given a symmetric key encryption scheme $(\mathsf{Enc}_S, \mathsf{Dec}_S)$ and a public key encryption scheme $(\mathsf{Gen}, \mathsf{Enc}_P, \mathsf{Dec}_P)$, the Fujisaki-Okamoto transformation is the tuple $(\mathsf{Gen}, \mathsf{Enc}_{FO}^{G,H}, \mathsf{Dec}_{FO}^{G,H})$ where:

- G, H are two functions, where G outputs keys for Enc_S and H outputs the random coins used by Enc_P.
- $\mathsf{Enc}_{FO}^{G,H}(\mathsf{pk}, m)$:
  - Choose a random input $\delta \in \{0,1\}^n$.
  - Compute $d \leftarrow \mathsf{Enc}_S(H(\delta), m)$.
  - Compute $c \leftarrow \mathsf{Enc}_P(\mathsf{pk}, \delta; G(\delta, d))$
  - Output (c, d)
- $\mathsf{Dec}_{FO}^{G,H}(\mathsf{sk}, (c, d))$:
  - Compute $\delta' \leftarrow \mathsf{Dec}_P(\mathsf{sk}, c)$
  - Check that $\mathsf{Enc}_P(\mathsf{pk}, \delta'; G(\delta', d)) = c$. If not, output $\bot$ and abort.
  - Compute and output $m' \leftarrow \mathsf{Dec}_S(H(\delta'), d)$

F.2 The Quantum CCA security of FO

We now prove the following theorem regarding the CCA security of the FO transformation:

Theorem 6. *If* $(\mathsf{Enc}_S, \mathsf{Dec}_S)$ *is one-time secure and* $(\mathsf{Gen}, \mathsf{Enc}_P, \mathsf{Dec}_P)$ *is well-spread and one-way secure, then* $(\mathsf{Gen}, \mathsf{Enc}_{FO}^{G,H}, \mathsf{Dec}_{FO}^{G,H})$ *is quantum CCA secure in the quantum random oracle model.*

Proof. We will prove security through a sequence of hybrid experiments. The proof is similar to the classical proof of security for the FO transformation, except that we will use compressed oracles in order to answer questions of the form "has the adversary queried on a particular input".

Let $\mathcal{A}$ be a quantum polynomial time adversary for the CCA security of the scheme. Consider the following hybrids.

Hybrid 0. This is the experiment $\texttt{CCA-RO-Exp}_0$, where m_0^* is encrypted during the challenge query. Let the challenge ciphertext be (c^*, d^*). Let the randomness for encryption be δ^*. Then, note that the function CCA can be written as:

$$CCA_0(c, d) = \begin{cases} \bot & \text{if the challenge query has happend, and } (c, d) = (c^*, d^*) \\ \mathsf{Dec}(\mathsf{sk}, (c, d)) & \text{otherwise} \end{cases}$$

Hybrid 1. This is identical to **Hybrid 0**, except that we now change the function $CCA(c, d)$ to be:

$$CCA_1(c, d) = \begin{cases} \bot & \text{if the challenge query has happend, and } c = c^* \\ \mathsf{Dec}^{G,H}(\mathsf{sk}, (c, d)) & \text{otherwise} \end{cases}$$

Lemma 40. *$\mathcal{A}$ cannot distinguish* **Hybrid 0** *from* **Hybrid 1***, except with negligible probability.*

Proof. Notice that the only difference between **Hybrid 0** and **Hybrid 1** is the definition of CCA, and that the function only differs on inputs of the form $(c^*, d), d \neq d^*$ where $\mathsf{Dec}(\mathsf{sk}, (c^*, d)) \neq \bot$. In particular, since decryption succeeds, it must be the case that

$$\mathsf{Enc}_P(\mathsf{pk}, \delta; G(\delta, d)) = c^* = \mathsf{Enc}_P(\mathsf{pk}, \delta^*; G(\delta^*, d^*))$$

for some string δ. But the correctness of Enc_P implies that $\delta = \delta^*$.

Now, let $G'(\delta, d) = \mathsf{Enc}_P(\mathsf{pk}, \delta^*; G(\delta, d))$. Notice that any differing input to CCA must collide with (δ^*, d^*). We invoke the following lemma:

Lemma 41 (Adapted from [BBBV97]). *Consider an adversary making q quantum queries to an oracle G. Suppose G is changed to G', and the adversary distinguishes this change with advantage ϵ. Then, if we measure a randomly chosen query of the adversary, with probability at least ϵ^2/q^2 the result will be a point x where $G(x) \neq G(x')$.*

Therefore, any distinguisher gives us a collision finder for G'. Moreover, once we fix δ^*, pk, we see that G' is a random function, except that the output distribution is non-uniform. By the well-spread property of Enc_P, we have that the output distribution of G' has super-logarithmic min-entropy. We can then invoke Balogh, Eaton, and Song [BES17], who show that for any polynomial number of queries to such a function, the probability of finding a collision is negligible. □

From this point forward, we will consider G as being implemented in the compressed standard oracle. Since this is equivalent to the uncompressed standard oracle, this does not affect the adversary's success probability.

We will now also make one additional change that does not affect the adversary: at the beginning of a CCA query, perform a test (in superposition) to see if (δ', d) is in the database for G, where $\delta' \leftarrow \mathsf{Dec}_P(\mathsf{sk}, c)$. Record the output of this test in an ancillary qubit. Then, immediately un-compute the test.

Hybrid 2. This is identical to **Hybrid 1** (with the modifications above), except we now move the un-computation of the test above until after we apply CCA_1.

Lemma 42. *$\mathcal{A}$ cannot distinguish* **Hybrid 1** *from* **Hybrid 2***, except with negligible probability.*

Proof. Notice that evaluating CCA_1 only interfaces with G by performing a test of whether $\mathsf{Enc}_P(\mathsf{pk}, \delta'; G(\delta', d)) = c$. This can be equivalently rephrased as testing if $G(\delta', d)$ lies within the set $S_{\delta',c}$ of random coins to make $\mathsf{Enc}_P(\mathsf{pk}, \delta')$ go to c. By the well-spread property of Enc_P, these random coins make a negligible fraction of all random coins. So we can apply Lemma 39 with $S = S_{\delta',c'}$ to conclude that flipping the order of tests is undetectable

Hybrid 3. This is identical to **Hybrid 2**, except that we now change CCA again. It will additionally take as input a bit b, which is the output of the test above; $b = 1$ if (δ', d) is in the database for G, and $b = 0$ otherwise. We define CCA_3 as:

$$CCA_3(b, c, d) = \begin{cases} \perp & \text{if the challenge query has happend, and } c = c^* \\ \perp & \text{if } b = 0 \\ \mathsf{Dec}^{G,H}(\mathsf{sk}, (c, d)) & \text{otherwise} \end{cases}$$

Lemma 43. *$\mathcal{A}$ cannot distinguish* **Hybrid 2** *from* **Hybrid 3***, except with negligible probability.*

Proof. We will change one query at a time from CCA_1 to CCA_3. Notice that if the adversary can distinguish the change with non-negligible probability, it's query must have non-negligible weight on c, d such that (1) $c \neq c^*$, (2) $b = 0$, and (3) $\mathsf{Enc}_P(\mathsf{pk}, \delta'; G(\delta', d)) = c$. But if $b = 0$, then $G(\delta', d)$ is actually in uniform superposition, so the probability of it satisfying (3) is negligible, by the well-spread property of Enc_P.

Hybrid 4. Notice that in **Hybrid 3** we perform $\delta' \leftarrow \mathsf{Dec}_P(\mathsf{sk}, c)$ three times: once to compute the test, once inside CCA_3, and once to un-compute the test. **Hybrid 4** will be identical to **Hybrid 3**, except that instead of computing δ' in this way, we will simply search for it in the database for G.

In particular, we first check if $c = c^*$; if so we set the output of CCA to $\perp$. Otherwise, we will scan over the inputs in the database for G, looking for inputs of the form (δ'', d). For each one, we will check if $\mathsf{Enc}_P(\mathsf{pk}, \delta''; G(\delta'', d)) = c$. If the check passes, we will set $\delta' = \delta''$ and stop the scan. Then we proceed to decrypt by computing $m' \leftarrow \mathsf{Dec}_S(H(\delta'), d)$, and set the output of CCA to be m'. If we do not find such a δ'', we will not set δ', and instead set the output of CCA to be $\perp$.

Lemma 44. *$\mathcal{A}$ cannot distinguish* **Hybrid 3** *from* **Hybrid 4**

Proof. First, if we ever set a δ' in **Hybrid 4**, then it must be the case by correctness of $(\mathsf{Gen}, \mathsf{Enc}_P, \mathsf{Dec}_P)$ that $\delta' = \mathsf{Dec}_P(\mathsf{sk}, c)$. Therefore, in **Hybrid 3**, we would have computed the correct δ', and then our test would have found (δ', d) in the database, so it would set $b = 1$. In this case, **Hybrid 3** would have set the output of CCA to be m'.

Similarly, if **Hybrid 3** would successfully decrypt, it must have been the case that (δ', d) was in the database. In this case, it would be found in **Hybrid 4**. Therefore, these two hybrids are identical.

Notice that in **Hybrid 4**, the decryption key sk is no longer needed.

Hybrid 5. This is identical to **Hybrid 4**, except that:

- We choose δ^* at the very beginning of the experiment
- On a query on superposition (δ, d) to G, we measure if $\delta = \delta^*$. If so, the experiment outputs a random bit and aborts. Otherwise, it continues as before.
- On a query on superposition δ to H, we measure if $\delta = \delta^*$. If so, the experiment outputs a random bit and aborts. Otherwise, it continues as before.

Lemma 45. *$\mathcal{A}$ cannot distinguish* **Hybrid 4** *from* **Hybrid 5***, except with negligible probability.*

Proof. If the adversary could distinguish the two hybrids, it must have a non-negligible query weight on inputs containing δ^*. Then we can measure a random query by the adversary, and obtain δ^* with non-negligible probability. This means that we can construct an efficient adversary which, given pk and the encryption of a random δ^*, can successfully decrypt c^* with non-negligible probability. This is a contradiction to the assumed security of Enc_P.

Hybrid 6. We further modify **Hybrid 5** and now compute the challenge ciphertext (c^*, d^*) as follows:

- Choose a random input $\delta^* \in \{0,1\}^n$, k^* in the key space of Enc_S, r^* in the space of random coins for Enc_P.
- Compute $d^* \leftarrow \mathsf{Enc}_S(k^*, m_0^*)$.
- Compute $c^* \leftarrow \mathsf{Enc}_P(\mathsf{pk}, \delta; r^*)$
- Output (c^*, d^*)

Note that this effectively sets $G(\delta^*, d^*) = r^*$ and $H(\delta^*) = k^*$. Since the adversary never queries G, H on these points these points, the values were uniformly random anyway. Therefore, this change is undetectable to the adversary.

Lemma 46. *$\mathcal{A}$ cannot distinguish* **Hybrid 5** *from* **Hybrid 6**

Hybrid 7. Finally, we change the challenge ciphertext from encrypting m_0^* to encrypting m_1^*.

Lemma 47. *$\mathcal{A}$ cannot distinguish* **Hybrid 6** *from* **Hybrid 7***, except with negligible probability.*

Proof. This follows from the security of Enc_S and the fact that k^* is independent of the adversary's view.

Hybrid 8-14. We now undo all the previous changes, one by one, keeping the challenge ciphertext as m_1^*. The proofs of indistinguishability are essentially identical.

By the time we get to **Hybrid 14**, we are in $\texttt{CCA-RO-Exp}_1$. Putting everything together, we have that $\texttt{CCA-RO-Exp}_0$ (**Hybrid 0**) is indistinguishable from $\texttt{CCA-RO-Exp}_1$ (**Hybrid 14**), thus proving the CCA security of the FO scheme.