# How to Record Quantum Queries, and Applications to Quantum Indifferentiability

Mark Zhandry
Princeton University

**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". Our central observation is that when viewing the adversary's query *and the oracle itself* in the Fourier domain, an oracle query switches from writing to the adversary's space to writing to the oracle itself. This allows a reduction to simulate the oracle by simply recording information about the adversary's query in the Fourier domain.

We then use this new technique to show the indifferentiability of 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 "post-quantum" security — security against quantum attack[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 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 classically 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 recording is impossible, by the quantum no-cloning theorem. 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 hard problem. Because of the difficulty in reading an adversary's query, it also becomes hard to adaptively program the random oracle, another very common 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 almost always never changes it afterward. 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] 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 $H$. In one case, $h$ is a random function, and $H$ is built from $h$ according to the hash function construction. In the other case, $H$ 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 cases.

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

[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]

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 returns, and $(x, y)$ is added to $D_0$.
- $h_1$ is more interesting. 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 $D_0$. 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, $h_0$ has the form

$$\sum_{x \in \{0,1\}^{2n}, u \in \{0,1\}^n} \alpha_{x,u} |x, u\rangle$$

In response, the oracle for $h_0$ will perform the map $|x, u\rangle \mapsto |x, u \oplus h_0(x)\rangle$. This will transform the query state to

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

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.

One natural way to try to remedy the issue is to not store a database for $h_0$ or $h_1$. This might be achieved by setting $h_0(x)$ to be a quantum pseudorandom function [Zha12a] evaluated at $x$. Such a function will be indisitnguishable from a truly random $h_0(x)$ [3]. This will certainly fix the issue above, but introduce 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 apparent in every construction of a hash function from a compression function. Indeed, it is easy to show the following: suppose there is a hash function $H : \{0,1\}^M \to \{0,1\}^N$ built from a compression function $h : \{0,1\}^m \to \{0,1\}^n$, and suppose it holds that $M + \log_2 N$ is slightly larger than $m + \log_2 n$. Then it must be that *any* simulator for *classical* indifferentiability cannot answer the $h$ queries with the same exact function each time. This is because, if the simulator answers the $h$ queries with the same function (denoted $S^H$ since the function makes $H$ queries) every time, then $h$ is in fact a function with a well-defined truth table of size $n2^m$. 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 $C^h = C^{S^H}$ 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 $S^H$, along with the $\epsilon$ fraction of of input/output pairs where $H$ and $C^{S^H}$ disagree. The total length of this compressed truth table is $n2^m + \epsilon 2^M(MN)$. As $\epsilon$ can be made arbitrarily (polynomially) small, and since $n2^m$ is smaller than $N2^M$, the compressed truth table can be made smaller than the original truth table for $H$, which has size $N2^M$.

[3] Zhandry [Zha12b] shows that actually, a $t$-wise independent function will suffice, for an appropriate choice of $t$

Therefore, any simulator must actually answer queries dependent on previous queries — in other words, it *must* record some information about the adversary's queries into its internal state. But the existing QROM techniques are utterly incapable of such recording. We therefore ask:

*Is it possible to build large hash functions from small compression functions such that the hash functions are indifferentiable from a quantum-accessible random oracle?*

The discussion above would seem to suggest the answer is no[4].

## 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 every has any information about the oracle, the *oracle must also have information about the adversary.* Therefore a simulator can always get away with recording *some* information about the adversary, if done carefully.

We then use the technique to prove the indifferentiability of the Merkle-Damqård 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], nad also gives very simple 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 $\sum_h |h\rangle$. This is because 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 adversary's state is mixed (since it depends on the random choice of $h$), and the superposition over $h$ is a *purification* of the adversary's mixed state.

Therefore, we will imagine the $h$ oracle as actually 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.

[4]In fact, the author was convinced for a long time that such domain extension was indeed impossible

Now, we will think of $h$ as a vector of length $2^m \times n$ by simply writing down all of the outputs of $h$ as a list. We will think of each $x, u$ pair as a 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

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

As mentioned earlier, we can view the oracle as a purification of the adversary's state, and the purification can be updated through the adversary's queries by following simple update rules.

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 implemented as above, 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 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. 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 $h(x)$.

**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 queries by the adversary, and what the value of the query 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 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 intuition appears to work, we need to make sure the simulator does not disturb the compressed oracle. Unfortunately, some disturbance is necessary. This is because testing whether an element is contained in $D_0$ amounts to a measurement in the Fourier domain; meanwhile, determining the value of $h_0(x)$ is a measurement in the primal. 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 implement the simulator where it only performs a polynomial number of measurements of the form "is $h_0(x) = y$". Indeed, the classical simulator can easily be implemented using such tests, and we just use the same implementation. Meanwhile, testing whether an element is contained in $D_0$ is simply a Fourier-domain test of the form "is $h_0(x) = 0$".

Now, these primal and Fourier measurements still do not commute. Fortunately, it is straightforward to show that they "almost" commute. 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 unfiform 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 measurements is always "almost" determined, meaning the measurement negligibly affects the state. This means that, no matter what initial state is, the two measurements "almost" commute.

Thus, the simulator can perform these measurements 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 we show this using a careful sequence of hybrids.

**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 the query $\sum_{x,u} |x, u\rangle$ in our example, when implemented as a phase query, simply becomes $\sum_x |x, 0\rangle$. Since $u = 0$, 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. After applying the query, the joint quantum system of the adversary and simulator becomes

$$\sum_x |x, u\rangle \sum_y |\{(x, y)\}\rangle (-1)^{y \cdot u}$$

Thus, the simulator can clearly tell that the adversary has queried on $x$. Later, when the adversary queries on the same state a second time, the query will erase the phase term. Then the simulator will actually remove the $(x, y)$ pair from it's database. This is because in the Fourier domain, the compressed Fourier oracle now contains $(x, 0)$, which gets removed (since we only include $(x, y)$ pairs with non-zero $y$). 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.

### 1.1.1 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 6, 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[5].

In contrast to our very simple proof, the prior 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, and may have further useful applications.

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 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.

This problem is usually called the $k$-SUM problem. However, $k$-SUM is typically described as a worst-case decisional problem, deciding whether a set of such points exists. This decisional problem is usually stated to have a query complexity of $\Theta(2^{m\frac{k}{(k+1)}})$ [BS13], where $m$ is the number of input bits. By setting $n = mk$ — which is the setting where a random function will have approximately a single $k$-SUM solution — we coincide with the existing lower bounds. The prior proof [BS13] actually only works in the setting where $n \gtrsim km$ and does not apply to settings where there are many $k$-SUM solutions. In contrast, our lower bound works for *any* setting of $m$ and $n$.

### 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.

### 1.3 Independent Work

Independently and concurrently, Carstens et al. [CETU18] also discuss the difficulties of quantum indifferentiability. They present a very similar argument for the difficulty of proving indifferentiability as we do. They then conclude that "quantum indifferentiability is probably impossible to achieve in many situations", and state a formal conjecture regarding such an impossibility. Finally, they prove concretely an impossibility for *perfect* simulation for certain protocols.

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

Our work demonstrates that Carstens et al.'s conjecture is false, as their conjecture would apply to Merkle-Damgård. Note also that our simulator is *im*perfect as it perturbs the state slightly with every query, and therefore the formal impossibility does not apply.

**Acknowledgments**

We would like to thank Jan Czajkowski, Ehsan Ebrahimi, Christian Majenz, Chistian Schaffner and Dominique Unruh for helpful comments. This work is supported in part by NSF. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

# 2 Preliminaries

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

## 2.1 Quantum

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.

## 2.2 Quantum Tests

We will often want to perform computations that look like measurements, but will not actually be a quantum measurement. We will call these operations quantum tests.

We will define two types of tests:

- Test if some registers $x$ are equal to some string $y$ in the computational basis. What we mean by this is that a new qubit is initialized to 0. Then, apply the unitary which maps $|y\rangle|b\rangle \to |y\rangle|b \oplus 1\rangle$ and $|x\rangle|b\rangle \to |y\rangle|b\rangle$ for $x \neq y$. The result of the test is not contained in the new qubit. When we want to uncompute the test, we simply perform the test a second time, and discard the qubit. Note that if notion happened between computation of the test and decomputation, then the new qubit will contain 0 and is un-entangled with the $x$ registers. However, other operations performed in between may cause the qubit to be entangled with $x$, so discarding the new qubit will result in a mixed state.

- Test if some registers $x$ are equal to some string $y$ in the Fourier domain. This is identical to the test above, except that we perform the Fourier transform to $x$ before and after applying the test.

The two types of tests above 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, and interleave the computation/decomputaiton steps, the result will not preserve the registers $x$. This is because the two measurements corresponding to these tests (namely, measuring the new qubit that is produced during computation of the test) do not commute. However, as we show below, they very nearly commute, meaning interleaving the tests, while guaranteed to modify the state $x$, only does negligibly.

**Lemma 2.1.** *Consider a potentially mixed quantum state $\rho$ on $n$ qubits. Let $S$ be a subset of $\{0,1\}^n$. Suppose the following are performed:*

- *Test if $|\psi\rangle$ is $y \in S$ in the computational basis.*
- *Test if $|\psi\rangle$ is $z$ in the Fourier basis.*
- *Un-compute the computational basis test.*
- *Un-compute the Fourier basis test.*

*Let $\rho'$ be the resulting state after these tests. Then the trace distance between $\rho$ and $\rho'$ is at most $24\sqrt{|S|/2^n}$*

*Proof.* We will prove the theorem for $z = 0$, the other cases handled almost identically, but with additional tracking of phase terms. The state of the algorithm consists of the original registers $|\psi\rangle$, together with two quibits $b, c$ for recording the results of the computational basis test and Fourier basis test, respectively. We will view $b, c$ in the Fourier domain. In this case, $b, c$ are intitaly in uniform superposition. For each computational basis test above, we have the that $|x\rangle|b\rangle|c\rangle$ obtains a phase $(-1)^{b\delta_x}$ where $\delta_x$ is 1 if $x \in S$ and zero otherwise. This can be rephrased as applying a conditional unitary to $|\phi\rangle$: if $b = 0$, we apply the identity, while if $b = 1$, we apply $\mathbf{I} - 2\sum_{x\in S}|x\rangle\langle x|$.

Similarly, each Fourier test can be described as a conditional unitary on $|\psi\rangle$: if $c = 0$, apply the identity, while if $c = 1$, we apply $\mathbf{I} - 2|w\rangle\langle w|$, where $|w\rangle = \frac{1}{\sqrt{2^n}}\sum_x |x\rangle$ is the uniform superposition.

Let $\mu$ be $\rho$, but applying the Fourier basis test and then the computational basis test. Let $\mu'$ be $\rho'$, but applying the Fourier basis test and then the computational basis test after the steps above. Equivalently, $\mu'$ is the result of taking $\rho$, and applying the computational basis test and then the Fourier basis test.

The trace distance between $\rho$ and $\rho'$ is at most the trace distance if we incorporate the new registers that record the results of the tests. This trace distance is in turn equal to the trace distance between $\mu$ and $\mu'$, since the tests are unitary.

The state $\mu$ can be written as:

$$\frac{1}{4}\sum_{b,b',c,c'} |b\rangle\langle b'| \otimes |c\rangle\langle c'| \otimes \Big((\mathbf{I} - 2c|w\rangle\langle w|)(\mathbf{I} - 2b\sum_{x\in S}|x\rangle\langle x|)\rho(\mathbf{I} - 2b'\sum_{x\in S}|x\rangle\langle x|)(\mathbf{I} - 2c'|w\rangle\langle w|)\Big)$$

whereas the state $\mu'$ can be written as:

$$\frac{1}{4}\sum_{b,b',c,c'}|b\rangle\langle b'|\otimes|c\rangle\langle c'|\otimes\Big((\mathbf{I}-2b\sum_{x\in S}|x\rangle\langle x|)(\mathbf{I}-2c|w\rangle\langle w|)\rho(\mathbf{I}-2c'|w\rangle\langle w|)(\mathbf{I}-2b'\sum_{x\in S}|x\rangle\langle x|)\Big)$$

Notice that $(\mathbf{I}-2c|w\rangle\langle w|)(\mathbf{I}-2b\sum_{x\in S}|x\rangle\langle x|)=\mathbf{I}-2b\sum_{x\in S}|x\rangle\langle x|-2c|w\rangle\langle w|+4bc\sum_{x\in S}\langle w|x\rangle|w\rangle\langle x|$ whereas $(\mathbf{I}-2b\sum_{x\in S}|x\rangle\langle x|)(\mathbf{I}-2c|w\rangle\langle w|)=\mathbf{I}-2b\sum_{x\in S}|x\rangle\langle x|-2c|w\rangle\langle w|+4bc\sum_{x\in S}\langle x|w\rangle|x\rangle\langle w|$. Notice that $\langle x|w\rangle=\langle w|x\rangle=\frac{1}{\sqrt{2^n}}$. Define

$$\begin{aligned}|z\rangle&=\frac{1}{\sqrt{|S|}}\sum_{x\in S}|x\rangle\\ U_{b,c}&=\mathbf{I}-2b\sum_{x\in S}|x\rangle\langle x|-2c|w\rangle\langle w|\\ V&=|z\rangle\langle w|\end{aligned}$$

If we take the difference $\mu-\mu'$, with some manipulations we get:

$$\frac{1}{4}\sum_{b,b',c,c'}|b\rangle\langle b'|\otimes|c\rangle\langle c'|\otimes\Big(4\sqrt{\frac{|S|}{2^n}}bc(V-V^\dagger)\rho U^\dagger_{b',c'}+4\sqrt{\frac{|S|}{2^n}}b'c'U_{b,c}\rho(V^\dagger-V)+\frac{16|S|}{2^n}bb'cc'(V\rho V^\dagger-V^\dagger\rho V)\Big)$$

Write $|\gamma_{b,c}\rangle=U_{b,c}\rho|z\rangle$ and $|\gamma'_{b,c}\rangle=U_{b,c}\rho|w\rangle$. Then this difference can be written as:

$$\begin{aligned}\sum_{b,b',c,c'}|b\rangle\langle b'|\otimes|c\rangle\langle c'|\otimes\Big(&\frac{1}{\sqrt{2^n}}(bc|z\rangle\langle\gamma'_{b',c'}|+b'c'|\gamma'_{b,c}\rangle\langle z|-bc|w\rangle\langle\gamma_{b',c'}|-b'c'|\gamma_{b,c}\rangle\langle w|)\\ &+\frac{4bb'cc'}{2^n}(\langle w|\rho|w\rangle|z\rangle\langle z|-\langle z|\rho|z\rangle|w\rangle\langle w|)\Big)\end{aligned}$$

Now, since $\langle w|\rho|w\rangle\leq 1$, $\langle 0|\rho|0\rangle\leq 1$, and $|\gamma_{b,c}\rangle,|\gamma'_{b,c}\rangle$ are vectors of norm at most 1, this difference can be written as a sum of 16 rank one matrices whose trace has an absolute value of at most $\sqrt{\frac{|S|}{2^n}}$, as well as 2 rank one matrices whose trace has an absolute value of at most $\frac{4|S|}{2^n}$. The trace norm of this matrix is therefore at most $16\sqrt{\frac{|S|}{2^n}}+2\frac{4|S|}{2^n}\leq 24\sqrt{\frac{|S|}{2^n}}$. The trace distance between $\mu$ and $\mu'$ is therefore at most this quantity as well. This completes the proof. □

# 3 Oracle Variations

Here, we describe several oracle variations.

**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 tuple $|x,y\rangle\otimes|H\rangle$. Here, $x,y$ is the query, and $H$ is the truth-table of the function, as above. It performs the map

$$|x,y\rangle\otimes|H\rangle\mapsto|x,y\oplus H(x)\rangle\otimes|H\rangle$$

We will call this oracle $\mathsf{StO}$.

**Phase Oracle.** This oracle takes as input a tuple $|x,z\rangle \otimes |H\rangle$. Here, $x,z$ is the query, and $H$ is the truth-table of the function. It performs the map

$$|x,z\rangle \otimes |H\rangle \mapsto (-1)^{y\cdot H(x)}|x,z\rangle \otimes |H\rangle$$

We will call this oracle $\mathsf{PhO}$. Notice that $\mathsf{PhO}$ and $\mathsf{StO}$ are equivalent by applying the Fourier transform to the $y$ registers. That is,

$$\mathsf{PhO} = (\mathsf{Id} \otimes \mathsf{H}^{\otimes n} \otimes \mathsf{Id}) \cdot \mathsf{StO} \cdot (\mathsf{Id} \otimes \mathsf{H}^{\otimes n} \otimes \mathsf{Id})$$

**Fourier Oracle.** This oracle takes as input a tuple $|x,z\rangle \otimes |D\rangle$, and $D$ is the truth-table of the a function. 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 $P_{x,z}(x') = \begin{cases} z & \text{if } x' = x \\ 0 & \text{if } x' \neq n \end{cases}$, and $D \oplus P_{x,z}$ is the function $(D \oplus P_{x,z})(x') = D(x') \oplus P_{x,z}(x')$.

We will call this oracle $\mathsf{FourierO}$. Notice that $\mathsf{FourierO}$ and $\mathsf{PhO}$ are equivalent by applying the Fourier transform to the $H$ registers. That is,

$$\mathsf{FourierO} = (\mathsf{Id} \otimes \mathsf{Id} \otimes \mathsf{H}^{\otimes n\times 2^m}) \cdot \mathsf{PhO} \cdot (\mathsf{Id} \otimes \mathsf{Id} \otimes \mathsf{H}^{\otimes n\times 2^m})$$

Notice that whether the Fourier or Phase oracle is simply a change of basis on the oracle side. This means it is completely inconsequential to the adversary which oracle is used.

Whether the oracle is implemented in the computational of Fourier domains is orthogonal to whether queries are made in the computational or phase domains. Therefore, we actually get four different oracle types: $\mathsf{StO}, \mathsf{PhO}, \mathsf{FourierStO}, \mathsf{FourierPhsO}$, where $\mathsf{FourierPhsO}$ is the $\mathsf{PhO}$ oracle described above, and $\mathsf{FourierStO}$ is the standard oracle, except that the oracle is represented in the Fourier domain.

**Compressed Fourier Oracle.** Typically, the oracle $H$ will be chosen at random. We can simulate this by initially setting the oracle registers to be the uniform superposition over all truth tables. In the Fourier domain, this corresponds to the all-zeros function. Therefore, when we implement the Fourier Oracle, the oracle will typically start off containing just $D = 0^{n2^m}$.

Notice furthermore that after $q$ queries, $D$ will be the sum of $q$ point functions — in particular, it will be zero in all but $q$ locations. Therefore, we can actually compress $D$ into an unordered list of at most $q$ pairs $(x,z)$ with distinct $x$ such that $z \neq 0$.

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 oracle $\mathsf{CFourierO}$. It starts with an empty database, and on each query it performs the map:

$$|x,z\rangle \otimes |D\rangle \mapsto |x,z\rangle \otimes |D \oplus (x,z)\rangle$$

where $D \oplus (x,z)$ is the procedure that does the following:

- If $z = 0$ it outputs $D$.

- If there is a pair $(x, z') \in D$ for some $z'$, it does the following:
  - If $z = z'$, it removes $(x, z)$ from $D$, and outputs the new $D$
  - if $z \neq z'$, it replaces $(x, z')$ with $(x, z \oplus z')$ and outputs the new $D$.
- Finally, if there is no such pair $(x, z') \in D$, it adds the pair $(x, z)$ to $D$ and outputs the new $D$.

Note that $D \oplus (x, z)$ is reversible, provided $z$ started out as zero as it always will in the compressed Fourier oracle. Thus, the operation above is unitary.

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. Thus we can actually get two compressed Fourier oracles $\mathsf{CFourierStO}$ and $\mathsf{CFourierPhsO}$. We will typically only use $\mathsf{CFourierO} = \mathsf{CFourierPhsO}$.

**Compressed Standard and Phase Oracles.** Finally, we can also compress the oracle in the computational basis, though more care is required. The compressed standard oracle will work by taking the compressed Fourier oracle, and simply performing the Fourier transform on all of the $z$ terms. Thus, the state of the Compressed Standard Oracle will be an unordered list $D$ of pairs $(x, y)$. Note here that $y$ can potentially be zero, but we still require that when we view $y$ in the Fourier domain, we see zero.

As always, this representation is independent of the adversary's view, and we can choose to give the adversary either a phase or standard oracle by appropriate change of basis on query registers. We will label these two oracles as $\mathsf{CStO}, \mathsf{CPhsO}$.

Note that in the compressed standard or phase oracle, each $(x, y)$ in $D$ pair corresponds to two statements: (1) $x$ has been queried by the adversary, and (2) the value of $H(x) = y$. Unlike the standard oracle in which we can measure $H(x)$ to get $y$, here we cannot measure the $y$ part of $x$ in the computational basis, as this will destroy the invariant that the $y$ registers must be non-zero in the Fourier domain (since a singleton state $y$ in the computational basis contains a 0 in the Fourier basis). In particular, if we perform such a measurement, we will not be able to correctly remove $(x, y)$ from $D$ in future queries if needed.

For completeness, we specify the Compressed Standard Oracle $\mathsf{CStO}$. Initially, the database $D$ is empty. To answer a query on $|x, y\rangle$, do the following:

- Look for a tuple $(x, y') \in D$. If one is found, respond with $|x, y \oplus y'\rangle$
- If no tuple is found, create new registers initialized to the state $\frac{1}{\sqrt{2^n}} \sum_{y'} |y'\rangle$. Add the registers $(x, y')$ to $D$. Then respond with $|x, y \oplus y'\rangle$.
- Finally, regardless of whether the tuple was found or added, there is now a tuple $(x, y')$ in $D$, which may have to be removed. To do so, test whether the registers containing $y'$ contain 0 in the Fourier basis. If so, remove the tuple from $D$. Otherwise, leave the tuple in $D$.

# 4 Quantum Query Lower 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 $0^n$. We will actually prove a potentially stronger statement, namely that:

**Theorem 4.1.** *After $q$ queries, if the compressed (standard or phase) oracle for $H$ is measured, the resulting database will contain a pair of the form $(x, 0)$ with probability at most $O(q^2/2^n)$*

Since the adversary only has information about the points in the compressed oracle, the only way for it to achieve a non-trivial success probability is to output an element in the compressed oracle's database. Theorem 4.1 bounds the probability such points can be a pre-image of zero.

*Proof.* We will prove the theorem for the compressed phase oracle, the compressed standard oracle following from the fact that they are equivalent by applying a unitary to the adversary's registers.

Clearly, at the beginning, the compressed oracle's database is empty, so the probability the database contains an $(x, 0)$ is 0. Let $0 \in D$ mean that $D$ contains a pair of the form $(x, 0)$. Let $D(x)$ be the function that outputs $y$ if there is a pair $(x, y)$ in $D$, and outputs $\perp$ if there is no such pair. Now, we will show that the probability cannot rise too much with each query. Consider the joint state of the adversary and oracle just before the $q$th 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 $D$ containing a 0. We will write $|\psi\rangle = |\psi_0\rangle + |\psi_1\rangle$ for two orthogonal unnormalized states $|\psi_0\rangle = P|\psi\rangle$ and $|\psi_1\rangle = (I - P)|\psi\rangle$.

Let $|\psi'\rangle$ be the result of applying the query and $|\psi_0'\rangle, |\psi_1'\rangle$ the the corresponding images of $|\psi_0\rangle, |\psi_1\rangle$. Let $|\psi_0''\rangle = P|\psi'\rangle, |\psi_1''\rangle = (I - P)|\psi'\rangle$ be the projections after the query. We are interested in bounding the norm of $|\psi_0''\rangle$. We observe that $|\psi_0''\rangle = P|\psi_0'\rangle + P|\psi_1'\rangle$, containing contributions from both $|\psi_0'\rangle$ and $|\psi_1'\rangle$. Denote these contributions as $|\psi_{0,0}'\rangle = P|\psi_0'\rangle$ and $|\psi_{0,1}'\rangle = P|\psi_1'\rangle$, respectively. Then we know that:

- $|\psi_0''\rangle = |\psi_{0,0}'\rangle + |\psi_{0,1}'\rangle$
- $\||\psi_{0,0}'\rangle\| \leq \||\psi_0\rangle\|$
- Let $|\phi\rangle$ be the subset of $|\psi_1\rangle$ for which the query registers $(x, y)$ have $y \neq 0$ and $D(x) = \perp$. Note that $|\psi_1\rangle - |\phi\rangle$ has only $y = 0$ or $D(x) \notin \{0, \perp\}$. In the $y = 0$ case, $D$ remains unchanged. In the $D(x) \notin \{0, \perp\}$, $D$ either remains unchanged or has the input $x$ removed. In either case, after the query $|\psi_1\rangle - |\phi\rangle$ still has $D$ *not* containing any $y \neq 0$. Thus, $|\psi_1\rangle - |\phi\rangle$ does not contribute to $|\psi_0''\rangle$.
- Consider applying the query to a basis state $|x, y, z\rangle \otimes |D\rangle$ in $|\phi\rangle$ (meaning $y \neq 0$ and $D(x) = \perp$). The basis state maps to

$$|x, y, z\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_w (-1)^{y \cdot w} |D \cup (x, w)\rangle$$

Note that since $D$ contains no zeros, when we apply the projection $P$, the only contribution of $|\phi\rangle$ to $|\psi'_0\rangle$ is

$$|x, y, z\rangle \otimes \frac{1}{\sqrt{2^n}}|D \cup (x, 0)\rangle$$

This means

$$\||\psi'_{0,1}\rangle\| = \frac{1}{\sqrt{2^n}}\||\phi\rangle\| \leq \frac{1}{\sqrt{2^n}}\||\psi_1\rangle\| = \frac{1}{\sqrt{2^n}}(1 - \||\psi_0\rangle\|)$$

- Putting everything together, we have that

$$\||\psi'_0\rangle\| \leq \||\psi_0\rangle\| + \frac{1}{\sqrt{2^n}}$$

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 is just the square of this norm, which is at most $\frac{q^2}{2^n}$.

□

## 4.2 Collision Lower Bound

We can also adapt the above proof for the Collision lower bound. If we define $P$ to project onto databases $D$ containing a collision, and re-define $|\psi_b\rangle, |\psi'_b\rangle, |\psi'_{0,b}\rangle, |\phi\rangle$ accordingly. Write $|\phi\rangle = \sum_{x,y\neq 0,z,D:D(x)=\perp} \alpha_{x,y,z,D}|x, y, z\rangle \otimes |D\rangle$.

Our goal is to bound $|\psi'_{0,1}\rangle = P|\psi'_1\rangle$. We have that if we project $|x, y, z\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_w (-1)^{y\cdot w}|D \cup (x, w)\rangle$ in $|\psi'_1\rangle$ onto $P$, we will get $|z, y, z\rangle \otimes \frac{1}{\sqrt{2^n}} \sum_{w\in D} |D \cup (x, w)\rangle\rangle$.

This allows us to write $\||\psi'_{0,1}\rangle\| = \||\phi'\rangle\|$ where

$$|\phi'\rangle = \sum_{x,y\neq 0,z,D:D(x)=\perp,w\in D} \alpha_{x,y,z,D}|x, y, z\rangle \otimes |D \cup (x, w)\rangle$$

Notice that $|\phi'\rangle$ is the sum of $|\phi_i\rangle$ for $i \in q$, where $w$ is set to be the $i$th image point in $D$. Notice that after $q$ queries, the total size of $D$ is at most $q$. Each of the $|\phi_i\rangle$ have norm at most 1, and have disjoint support, so $\||\phi\rangle\| \leq \sqrt{q}$.

Applying the above arguments, this means the norm of $|\psi_0\rangle$ increases by at most $\frac{\sqrt{q}}{\sqrt{2^n}}$. Therefore, after $q$ queries, the total norm of $|\psi_0\rangle$ is at most $\frac{q^{3/2}}{\sqrt{2^n}}$. Thus, the probability that $D$ contains a collision is at most $q^3/2^n$.

Then we get the theorem:

**Theorem 4.2.** *After $q$ queries, if the compressed (standard or phase) oracle for $H$ is measured, the resulting database will contain a collision with probability at most $O(q^3/2^n)$*

## 4.3 More General Settings

The above proof technique is very general. Suppose we have a relation $R$ such that, for any input $x$, if $R$ is not satisfied on a database $D$ of size $q$, then there are at most $k(q)$ possible pairs $(x, w)$ that can be added to $D$ to make $R$ satisfied. Then we have that the total probability $R$ is satisfied on $D$ after $q$ queries is at most $(\sum_{i=1}^q \sqrt{k(q)})^2/2^n$.

This can be used to easily show optimal bounds for the $k$-sum problem for a random oracle:

**Theorem 4.3.** *After $q$ queries to a random oracle, $D$ will contain $k$ distinct tuples $(x_i, y_i)$ such that $\sum_i y_i = 0$ with probability at most $O(q^{k+1}/2^n)$, matching the optimal algorithm.*

# 5 Quantum Indifferentiability of Merkle-Damgård

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

## 5.1 Background

**Pre-fix free encoding.** A prefix-free code over $\{0,1\}^*$ is a set $S$ such that, for all $x \neq y \in S$, $x$ 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 $S$ 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 $z_1 = w_1$.
- For each $i = 2, \dots, \ell$, let $z_i = h(z_{i-1}, w_i)$.
- Output $y_\ell$.

**Quantum Indifferentiability.** 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 circuit that makes oracle queries to $h$.

**Definition 5.1.** $C^h$ is quantum indifferentiable from a random oracle $H : \{0,1\}^M \to \{0,1\}^N$ if, for any polynomial-time distinguisher $D$, there exists a polynomial-time simulator $S$ such that $S$ makes queries to $H$ and:

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

Here, $D$ is given quantum oracle access to $h$ and $H$. On the left hand size, $H = C^h$ and $h$ is a random oracle, while on the right-hand side $h = S^H$ and $H$ is a random oracle.

It is also straightforward to adapt this definition to handle the case of many random compression functions $h_1, \dots, h_n$.

## 5.2 Quantum Indifferentiability of Merkle-Damgård

We now prove the quantum indifferentiability of Merkle-Damgård, by exhibiting a simulator as required by Definition 5.1. We consider an adversary interacting with $h, H$. In the real world, $h$ is a uniformly random function, and $H = MD_h$. In the ideal world, $H$ is chosen uniformly at random, and we must construct a simulator $Sim$ for $h$. We must show that these two worlds are indistinguishable.

Our simulator is defined as follows. The simulator implements $h$ as the compressed standard oracle $\mathsf{CStO}$, but will make occasional exceptions in order to make sure the oracle is "consistent with" $H$. $Sim$ maintains an unordered database $D$ of $(x, y)$ pairs, in superposition. $D$ is initially empty.

First, after every query, $Sim$ will check that $D$ contains no collision in $D$: two pairs $(x_0, y_0), (x_1, y_1)$ such that $y_0 = y_1$. That is, it will initialize an auxiliary qubit, and flip the bit if there is a collision in $D$. Then it measures the qubit. If the result is 1, $Sim$ immediately aborts. Otherwise, the qubit contains 0, so it can be discarded and $Sim$ continues. This ensures that the state of $D$ never contains collisions.

We describe how $Sim$ operates on basis states $|x, y\rangle$, though $Sim$ will actually operate in superposition. $Sim$ first looks for a "completion of" $x$ in $D$. A completion is defined in the same was as in the classical case. A completion is a list of entries $((z'_{i-1}, w_i), z_i) \in D$ for $i = 2, \dots \ell - 1$ such that, if we write $x = (z'_{\ell-1}, w_\ell)$:

- $z'_i = z_i$
- If we let $w_1 = z_1$, then $(w_1, \dots, w_\ell)$ is in the prefix-free code $S$.

If $Sim$ finds a single completion, let $w = (w_1, \dots, w_\ell)$. Then $Sim$ will make the query $|w, y\rangle$ to the oracle $H$ by splicing together the $w$ it just computed with the $y$ registers from the adversary's query. The response will be $|w, y \oplus H(w)\rangle$. Then it uncomputes the completion and $w$, and finally returns $|x, y \oplus H(w)\rangle$ to the adversary by reconstituting the $x$ and $y$ registers from the original query.

The fact that there are no collisions in $D$ and that $S$ is a prefix-free code implies that there is only at most a single completion, so we do not need to worry about what happens if $Sim$ finds multiple completions (though for concreteness, imagine that $S$ chooses the first such completion).

If $Sim$ finds no completions, then it proceeds as if it was just the oracle $\mathsf{CStO}$ (it does not perform a $\mathsf{CStO}$ operation in either of the above two cases).

**Security Analysis.** We now prove the efficacy of our simulator.

**Theorem 5.2.** *For any distinguisher $A$ making at most $q$ queries, $A$ cannot distinguish the real from ideal world except with probability $O(q^4 2^{-n/2})$*

*Proof.* We will prove security using a sequence of hybrids.

**Hybrid 0.** This hybrid is the real world where $h$ is random and $H$ is implemented as $\mathsf{MD}_h$. Here, we will think of $h$ as implemented using the compressed standard oracle $\mathsf{CStO}$ with database $D$.

**Hybrid 1.** In this hybrid, we will implement $H$ as $\mathsf{MD}_h$, but we will tweak the implementation of $h$. After every query, $h$ will look at its database $D$, and search for a collision: two tuples $(x, y), (x', y')$ such that $x \neq x'$ but $y = y'$. It will initialize an auxiliary bit to 0. If it finds any such collision, it will flip the bit to 1.

Then, it measures the auxiliary bit. If the bit is 0, it discards the bit and continues. Otherwise, if the bit is 1 it will immediately abort.

By the collision bound (Theorem 4.2), the probability that $D$ contains a collision is negligible, so the probability of abort is negligible. This means the measurement negligibly affects the state.

**Hybrid 2.** In this hybrid, we will change $h$ again, while keeping $H$ as $\mathsf{MD}_h$. Here, instead of storing the state for $h$ as a single database $D$, we will store the state in "encoded" form as two databases $D, E$. To process each query, we will decode, apply $h$ as in **Hybrid 1**, and then re-encode.

The encoding is an entirely classical procedure, but applied in superposition to the database $D$. To encode, start with an empty $E$ and do the following. First, for each tuple $(x, y) \in D$, test if $(x, y)$ has a completion amongst the remaining tuples of $D$. If a completion is found for $(x, y)$ with string $w$, then add the tuple $(w, y)$ to $E$. We need this operation to be reversible, so we actually toggle whether $(w, y)$ is in $E$. If no completion is found for $(x, y)$, do nothing.

Then for each $(w, y) \in E$, test if $(w, y)$ corresponds to some tuple $(x, y) \in D$ that has a completion in $D$. Concretely, do the following: first set $z_1 = w_1$. Then, for $i = 2, \dots \ell - 1$, test if there is a tuple $((z_{i-1}, w_i), z_i) \in D$. If not, stop, and report that no completion is found. Otherwise continue. Finally, let $x = (z_{\ell-1}, w_\ell)$, and report that there is a tuple with a completion.

If so, remove $(x, y)$ from $D$ (as before, we want this to be reversible, so we toggle whether $(x, y)$ is in $D$). Otherwise, do nothing.

We note that, since by **Hybrid 1** $D$ has no collisions and since $S$ is a prefix-free code, we have that:

- Any tuple $(x, y)$ has at most one completion
- If $(x, y)$ has a completion, then $(x, y)$ is not a part of any other completion
- No two completions correspond to the same $w$
- This means that it does not matter which completion we use (since there will only be one), or the order in which we process completions
- This also means that whenever we are processing a $(w, y)$, there will be exactly one pair $(x, y)$ corresponding to $w$ that has a completion in $D$.
- This finally means that encoding is a reversible process. To decode, iterate over all $(w, y)$ in $E$. Search for the completion corresponding to $w$ (it is guaranteed to exist and be unique). Write the corresponding $(x, y)$ tuple to $D$. After processing all $(w, y) \in E$, iterate over all $(x, y)$ and see if $(x, y)$ has a completion with string $w$. If so, remove $(w, y)$ from $E$.

The initial state of $h$ is $D, E$ are both empty, which corresponds to encoding an empty $D$.

**Hybrid 3.** In this hybrid, we will change $h$ again, while keeping $H$ as $\mathsf{MD}_h$. Here, however, we will modify the encoding/decoding procedure above slightly. The intuition is that we remove an $(x, z)$ pair that had a completion, we potentially can remove the pairs in the completion as well to be consisent with the Compress Standard Oracle.

During encoding, consider processing a pair $(w, y) \in E$, which resulted in removing some pair $(x = (z_{\ell-1}, w_\ell), y)$ from $D$. Let $\{((z_{i-1}, w_i), z_i)\}_{i=2,\dots\ell-1}$ in $D$ be the completion of $(x, y)$. After removing $(x, y)$, we then do the following for $i = \ell - 1, \dots, 2$:

- Take the pair $((z_{i-1}, w_i), z_i)$, and test if $z_i$ is 0 in the Fourier domain.
- If $z_i$ is 0 in the Fourier domain, then remove the pair $((z_{i-1}, w_i), z_i)$ from $D$
- If $z_i$ is non-zero in the Fourier domain, stop and move on to the next $(w, y)$ pair in $E$

It is important to process the completion in reverse order, since if $((z_{i-1}, w_i), z_i) \in D$, then almost certainly measuring $((z_{i-2}, w_{i-1}), z_{i-1})$ in the Fourier basis will result in non-zero, since $z_{i-1}$ is still in use.

Similarly, during decoding, when processing a $(w, y) \in E$, if we would abort the completion finding process, instead gradually create a completion in $D$. That is, when testing if there is a tuple $((z_{i-1}, w_i), z_i) \in D$, if we don't find one, we instead do the following: for $j = i, \dots, \ell-$: create a tuple $((z_{j-1}, w_j), z_j)$ where $z_j$ is the uniform superposition over all possible $z_j$ and add this tuple to $D$. Finally, let $x = (z_{\ell-1}, w_\ell)$, and report that there is a tuple with a completion.

Notice that if we ever remove a tuple from $D$ during encoding, we will be guaranteed to add it back in during decoding. Therefore, the only difference between **Hybrid 2** and **Hybrid 3** is that introduce some extra tests (namely, testing whether a pair is 0 in the Fourier domain). However, we claim that these tests have negligible effect on the state. Indeed, define $P_{x,y}$ as the test that looks up the pair $(x, y') \in D$ and (provided it exists), outputs 1 if and only if $y' = y$. Similarly, define $T_x$ as the test that looks up the pair $(x, y') \in D$, and (provided it exists), outputs 1 if and only if $y'$ is 0 in the Fourier domain.

Lemma 2.1 show that $T_x$ and $P_{x,y}$ *almost* commute. Moreover, for every point $x$ added to $D$, the measurement $T_x$ is already applied at least once when $x$ is first added. Finally, it is straightforward to implementing all of the testing for completions as a sequence of polynomially-many applications of $P_{x,y}$. Piecing this all together, we can move every $T_x$ measurements performed in **Hybrid 3** to occur exactly when $x$ is added to $D$, and then absorb it with the $T_x$ that already exists there. By the near commutativity of $T_x$ and $P_{x,y}$, this incurs only a negligible affect on the final state.

**Hybrid 4.** Notice that in **Hybrid 3**, if we pretended that $T_x$ and $P_{x,y}$ actually commuted, the ultimate effect of decoding, handling the query $|x, y\rangle$, and then re-encoding is that most of the $D, E$ tuples will be unaffected. The only tuples that may be affected are $(x, y) \in D$, or the tuple $(w, y) \in E$ if $w$ corresponds to a completion of $x$.

Therefore, in **Hybrid 4**, we do not decode or encode anything except tuples relating to $|x, y\rangle$. By the near commutativity of $T_x$ and $P_{x,y}$, this will only negligibly affect the state.

At this point, an equivalent way of describing **Hybrid 4** is as follows. We implement $h$ as $Sim$, which makes queries to an oracle $H'$. The state of $Sim$ will be $D$, and $H'$ will be implemented using the compressed standard oracle with database $E$. $H$ will still be $\mathsf{MD}_h$.

**Hybrid 5.** Finally, we describe the ideal world, which is the same as **Hybrid 4**, except that we set $H$ to be the same as $H'$. $h$ is still implemented as $Sim$, which makes oracle queries to $H' = H$.

We now verify that **Hybrid 4** and **Hybrid 5** are identical. Since $h$ is implemented the same way in both hybrids, we only need to verify that a query to $\mathsf{MD}_h$ is identical to making a query directly to $H'$.

Indeed, it is straightforward to verify that after making the initial $\ell - 1$ calls to $h$, the final query will detect a completion, resulting in the desired query to $H'$. Moreover, it is straightforward to verify that any effect on $h$ the queries leading up to the final call have will be uncomputed when $\mathsf{MD}_h$ un-computes its intermediate values.

We can make all of the above steps quantitative, using the following lemma of [BBBV97]:

**Lemma 5.3** ([? ]). *Let $|\varphi\rangle$ and $|\psi\rangle$ be quantum states with Euclidean distance at most $\epsilon$. Then, performing the same measurement on $|\varphi\rangle$ and $|\psi\rangle$ yields distributions with statistical distance at most $4\epsilon$.*

The only transitions that cause any change are:

- **Hybrid 0** to **Hybrid 1**. Here, the error caused by each step is a vector of norm $O(\sqrt{q}2^{-n/2})$. The overall error is therefore $O(q^{3/2}2^{-n/2})$
- **Hybrid 2** to **Hybrid 3**. Here, we can implement the completion check using $q^2$ primal-domain tests. Re-ordering these tests with the Fourier domain tests results in an error of $O(q^2 2^{-n/2})$ for each query by Lemma 2.1.
- **Hybrid 3** to **Hybrid 4**. Here, the only difference is re-ordering up to $q$ tests in the Fourier domain with the up to $q^2$ primal tests. This results in an error of $O(q^3 2^{-n/2})$ for each query by Lemma 2.1. The overall error is $O(q^4 2^{-n/2})$

Piecing everything together, we find that the total error is at most $O(q^4 2^{-n/2})$.

□

# 6 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].

## 6.1 Background

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 $\lambda$, 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 $\mathsf{pk}$ and a message $m$, and produces a ciphertext $c$
- $\mathsf{Dec}(\mathsf{sk}, c)$ takes as input a secret key $\mathsf{sk}$ 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 $\mathsf{pk}$ 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 $\lambda$. 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 $\mathsf{pk}$ 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 $\mathsf{pk}$ 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
      $$CCA(c) = \begin{cases} \perp & \text{if } c \text{ was the result of a previous challenge query} \\ \mathsf{Dec}(\mathsf{sk}, c) & \text{otherwise} \end{cases}$$
      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 $\mathsf{Enc}_S$ and $H$ outputs the random coins used by $\mathsf{Enc}_P$.
- $\mathsf{Enc}_{FO}^{G,H}(\mathsf{pk}, m)$:
  - 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 $\perp$ and abort.
– Compute and output $m' \leftarrow \mathsf{Dec}_S(H(\delta'), d)$

## 6.2 The Quantum CCA security of FO

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

**Theorem 6.1.** *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 $\delta^*$. Then, note that the function $CCA$ can be written as:

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

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

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

**Lemma 6.2.** $\mathcal{A}$ *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 $\mathsf{Dec}(\mathsf{sk}, (c^*, d)) \neq \perp$. 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 $\delta$. But the correctness of $\mathsf{Enc}_P$ 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 $(\delta^*, d^*)$. We invoke the following lemma:

**Lemma 6.3** (Adapted from [**?** ])**.** *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* $\epsilon$. *Then, if we measure a randomly chosen query of the adversary, with probability at least* $\epsilon^2/q^2$.

Therefore, any distinguisher gives us a collision finder for $G'$. Moreover, once we fix $\delta^*, \mathsf{pk}$, we see that $G'$ is a random function, except that the output distribution is non-uniform. By the well-spread property of $\mathsf{Enc}_P$, 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: in a CCA query, perform a test (in superposition) to see if $(\delta', d)$ has previously been queried to $G$, 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 6.4.** *$\mathcal{A}$ 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 $\mathsf{Enc}_P$, these random coins make a negligible fraction of all random coins. So we can apply Lemma 2.1 with $S = S_{\delta',c'}$ to conclude that flipping the order of tests is undetectable $\square$

**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 check above; $b = 1$ if the adversary had queried on $(\delta', d)$, 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 6.5.** *$\mathcal{A}$ 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 $\mathsf{Enc}_P$. $\square$

**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 $\delta'$ 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 $(\delta'', d)$. 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 $\delta''$, we will not set $\delta'$, and instead set the output of $CCA$ to be $\perp$.

**Lemma 6.6.** *$\mathcal{A}$ cannot distinguish* **Hybrid 3** *from* **Hybrid 4**

*Proof.* First, if we ever set a $\delta'$ 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 $\delta'$, and then our test would have found $(\delta', d)$ 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 $(\delta', d)$ 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 $\mathsf{sk}$ is no longer needed.

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

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

**Lemma 6.7.** *$\mathcal{A}$ 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 $\delta^*$. Then we can measure a random query by the adversary, and obtain $\delta^*$ with non-negligible probability. This means that we can construct an efficient adversary which, given $\mathsf{pk}$ and the encryption of a random $\delta^*$, can successfully decrypt $c^*$ with non-negligible probability. This is a contradiction to the assumed security of $\mathsf{Enc}_P$. □

**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 $\mathsf{Enc}_S$, $r^*$ in the space of random coins for $\mathsf{Enc}_P$.
- 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 6.8.** *$\mathcal{A}$ cannot distinguish* **Hybrid 5** *from* **Hybrid 6**

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

**Lemma 6.9.** *$\mathcal{A}$ cannot distinguish* **Hybrid 6** *from* **Hybrid 7***, except with negligible probability.*

*Proof.* This follows from the security of $\mathsf{Enc}_S$ 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. □

## References


[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, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

[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, Germany, 1998.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[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, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 361–379, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[CETU18] Tore Vincent Carstens, Ehsan Ebrahimi, Gelo Noel Tabia, and Dominique Unruh. On quantum indifferentiability. Cryptology ePrint Archive, Report 2018/257, 2018. https://eprint.iacr.org/2018/257.

[DFG13] Özgür Dagdelen, Marc Fischlin, and Tommaso Gagliardoni. The Fiat-Shamir transformation in a quantum world. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part II*, volume 8270 of *Lecture Notes in Computer Science*, pages 62–81, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554, Santa Barbara, CA, USA, August 15–19, 1999. Springer, Heidelberg, Germany.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on Theory of Computing*, pages 212–219, Philadephia, PA, USA, May 22–24, 1996. ACM Press.

[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, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 603–621, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[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: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[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, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506, Tallinn, Estonia, May 15–19, 2011. Springer, Heidelberg, Germany.

[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: 14th Theory of Cryptography Conference, Part II*, volume 9986 of *Lecture Notes in Computer Science*, pages 192–216, Beijing, China, October 31 – November 3, 2016. Springer, Heidelberg, Germany.

[Unr15] Dominique Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 755–784, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[Unr16] Dominique Unruh. Collapse-binding quantum commitments without random oracles. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 166–195, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[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 Annual Symposium on Foundations of Computer Science*, pages 679–687, New Brunswick, NJ, USA, October 20–23, 2012. IEEE Computer Society Press.

[Zha12b] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 758–775, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

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