Secure Identity-Based Encryption in the Quantum Random Oracle Model

Mark Zhandry
Stanford University, USA
mzhandry@stanford.edu

Abstract

We give the first proof of security for an identity-based encryption scheme in the *quantum* random oracle model. This is the first proof of security for *any* scheme in this model that requires no additional assumptions. Our techniques are quite general and we use them to obtain security proofs for two random oracle hierarchical identity-based encryption schemes and a random oracle signature scheme, all of which have previously resisted quantum security proofs, even using additional assumptions. We also explain how to remove the extra assumptions from prior quantum random oracle model proofs. We accomplish these results by developing new tools for arguing that quantum algorithms cannot distinguish between two oracle distributions. Using a particular class of oracle distributions, so called *semi-constant* distributions, we argue that the aforementioned cryptosystems are secure against quantum adversaries.

Keywords: Quantum, Random Oracle, IBE, Signatures

1 Introduction

While quantum computation is not yet viable, Shor [Sho97] showed that when fully realized, quantum computers will break most of the cryptosystems used today, namely those based on the difficulty factoring and the discrete log problem. This has sparked the field of post-quantum cryptography, the search for classical systems secure against quantum adversaries. To be secure in this setting, a system must have an underlying difficult problem for quantum computers, as well as a security reduction showing how to solve this problem using a quantum adversary that breaks the system. Problems based on lattices have so far resisted quantum attacks, and there is a considerable amount of literature on lattice-based cryptosystems. However, random oracle security proofs for these constructions typically only consider classical adversaries, thus failing to show security in the quantum world.

The random oracle model [BR93] is one area where many classical proofs lack quantum equivalents. This model is of interest because random oracle schemes tend to be more efficient than their standard model counterparts. Consequently, the most efficient lattice-based schemes are often constructed in the random oracle model. For example, Gentry, Peikert, and Vaikuntanathan [GPV08] show how to construct signatures and identity-based encryption (IBE). Cash et al. [CHKP10] and Agrawal, Boneh, and Boyen [ABB10] give hierarchical IBE schemes both in the standard model and the random oracle model, with the random oracle constructions being more efficient than the corresponding standard model schemes. Gordon, Katz, and Vaikuntanathan [GKV10] give a group signature

scheme, while Boneh and Freeman [BF11] demonstrate a homomorphic signature scheme, all in the random oracle model.

All of these papers prove security against adversaries with classical access to the random oracle. However, when the scheme is instantiated, the random oracle is replaced with a hash function H, which a quantum adversary may evaluate on a quantum superposition of inputs. To model this ability, it is necessary to allow a quantum adversary to make quantum queries to the random oracle. We call this model the quantum random oracle model.

Proving security in the quantum random oracle model presents many challenges. Among them is the difficulty of efficiently simulating the random oracle. In the classical case, the random oracle is simulated on the fly, only generating randomness as needed. In the quantum setting, the adversary can query the oracle on an exponential superposition of inputs. Therefore, to make even the first query look random to the adversary, it would seem that we would need exponential randomness.

In addition to this difficulty, there are classical random oracle techniques that do not make sense if the adversary has quantum access to the random oracle. One such technique is that of Bellare and Rogaway [BR93] for proving the security of the Full Domain Hash signature scheme. In this technique, the reduction algorithm is given a challenge c to solve, and randomly guesses which oracle query the adversary will use to break the scheme. The algorithm embeds c into the response for this query, and if both the guess is correct and the adversary breaks the scheme, then the algorithm will be able to solve c.

In the quantum setting, this argument no longer applies because each oracle query might be over a superposition of exponentially many inputs. We could choose a random query and plug c into all outputs for that query, but this will not look like a random oracle to the adversary. Alternatively, we could choose one oracle input and plug c into the corresponding output for all queries. However, our chance of guessing correctly will then be exponentially small.

1.1 Our Contributions

We resolve some of the issues outlined above by giving a quantum analog of the technique of Bellare and Rogaway [BR93] and demonstrating how to simulate a random oracle without any additional computational assumptions. Specifically, we:

- Describe new ways to argue that quantum algorithms cannot distinguish between two distributions of oracles.
- Apply these techniques to a new type of distribution of oracles, which we call *semi-constant* distributions, showing that they cannot be distinguished from random oracles.
- Use our results on semi-constant distributions to prove that the random oracle IBE scheme of Gentry et al. [GPV08] is secure against quantum adversaries. The basic idea is to plug the challenge c into a small fraction of inputs to the oracle, making the oracle seen by the adversary a semi-constant distribution. The adversary thus behaves as though the oracle is random. If the adversary happens to use any of the inputs in this fraction, we are able to solve c.
- Show that this technique is general by applying it to the random oracle hierarchical IBE schemes of Cash et al. [CHKP10] and Agrawal et al. [ABB10].
- Prove that the generic Full Domain Hash signature scheme [BR93] is secure against quantum adversaries, though this remains a theoretical result until a trapdoor permutation is found that is secure against quantum adversaries.

- Use our techniques and k-wise independent functions to solve the problem of simulating quantum random oracles, thus making the above results unconditional.

1.2 Related Work

Quantum random oracles have been used in several prior works. For example, Bennett et al. [BBBV97] prove several quantum complexity results, including a proof that quantum computers cannot solve all of NP, relative to a random oracle. In a cryptographic setting, quantum random oracles have been used by Aaronson [Aar09] to construct quantum money and by Brassard and Salvail [BS08] and Brassard et al. [BHK+11] to construct quantum analogs of Merkle's Puzzles.

Boneh et al. [BDF+11] give a random oracle scheme that is secure against quantum adversaries with classical access to the oracle, but is insecure once the adversary has quantum access to the oracle. Despite this result, they show that there are circumstances in which a classical random oracle security reduction can also be used in the quantum setting. However, their proof techniques do not apply to the schemes analyzed in this paper. Additionally, their proof simulates the random oracle using a pseudorandom function (PRF) that is secure against quantum adversaries, hence making their results conditional on the existence of a so-called quantum-secure PRF. We note that Zhandry [Zha12] shows that such PRFs can be built from quantum-secure pseudorandom generators, which can, in turn, be built from lattices.

There has also been progress toward converting classical security proofs into quantum proofs outside of the random oracle model. For instance, Unruh[Unr10] shows that classical statistical security in Canetti's universal composabitily (UC) model implies quantum statistical security. Hallgren, Smith, and Song [HSS11] give a two-party protocol that is quantum computationally secure in the UC.

2 Preliminaries

A function $\epsilon(n)$ is negligible if it is non-negative and smaller than any inverse polynomial. That is, for any polynomial $p(n)$, $\epsilon(n) < 1/p(n)$ for all sufficiently large n.

A probabilistic polynomial time (PPT) algorithm is a classical randomized algorithm that runs in time polynomial in the size of its input. We also call such algorithms efficient.

2.1 Weight Assignments

A weight assignment on a set $\mathcal{X}$ is a function $D : \mathcal{X} \to \mathbb{R}$ such that $\sum_{x\in\mathcal{X}} D(x) = 1$. We sometimes write $\Pr_D[event]$ to represent the sum of the weights of all outcomes consistent with that event. A distribution on X is a weight-assignment D such that $D(x) \geq 0$ for all $x \in \mathcal{X}$. If D is a distribution, way that x occurs with probability $D(x)$. Let $U_{\mathcal{X}}$ denote the uniform distribution over $\mathcal{X}$. That is, $U_{\mathcal{X}}(x) = 1/|\mathcal{X}|$. When the set $\mathcal{X}$ is clear, we may omit the subscript.

We define the distance between two weight assignments D_1 and D_2 over a set $\mathcal{X}$ as

$$|D_1 - D_2| = \sum_{x\in\mathcal{X}} |D_1(x) - D_2(x)|$$

If $|D_1 - D_2| \leq \epsilon$, we say that D_1 and D_2 are ϵ-close.

Given a set of weight assignments D_y over $\mathcal{X}$, indexed by $y \in \mathcal{Y}$, and a weight assignment D over $\mathcal{Y}$, we can define a weight assignment D' over $\mathcal{X}$ where

$$D'(x) = \sum_{y \in \mathcal{Y}} D(y) D_y(x)$$

We write $D' = \sum_{y \in \mathcal{Y}} D(y) D_y$. Say each of the D_ys are actually distributions. Given two weight assignments D_1 and D_2 over $\mathcal{Y}$, define $D_1' = \sum_{y \in \mathcal{Y}} D_1(y) D_y$ and $D_2' = \sum_{y \in \mathcal{Y}} D_2(y) D_y$. Then:

$$\begin{aligned} |D_1' - D_2'| &= \sum_{x \in \mathcal{X}} \left| \sum_{y \in \mathcal{Y}} (D_1(y) - D_2(y)) D_y(x) \right| \\ &\leq \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} |(D_1(y) - D_2(y)) D_y(x)| \\ &= \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} |D_1(y) - D_2(y)| D_y(x) \\ &= \sum_{y \in \mathcal{Y}} |D_1(y) - D_2(y)| = |D_1 - D_2| \end{aligned}$$

Consider the set of functions $H : \mathcal{X} \to \mathcal{Y}$ for sets $\mathcal{X}$ and $\mathcal{Y}$, denoted by $\mathcal{H}_{\mathcal{X},\mathcal{Y}}$. Consider a weight assignment D on $\mathcal{H}_{\mathcal{X},\mathcal{Y}}$. Let $\mathcal{W} \subseteq \mathcal{X}$. We define the marginal weight assignment $D_{\mathcal{W}}$ of D on $\mathcal{H}_{\mathcal{W},\mathcal{Y}}$ where the weight of a function $H_{\mathcal{W}} : \mathcal{W} \to \mathcal{Y}$ is equal to the sum of the weights of all $H \in \mathcal{H}_{\mathcal{X},\mathcal{Y}}$ that agree with $H_{\mathcal{W}}$ on $\mathcal{W}$. In other words,

$$D_{\mathcal{W}}(H_{\mathcal{W}}) = \Pr_D[H(w) = H_{\mathcal{W}}(w) \forall w \in \mathcal{W}]$$

We call two weight assignments D_1 and D_2 on $\mathcal{H}_{\mathcal{X},\mathcal{Y}}$ k-wise equivalent if for all $\mathcal{W} \subseteq \mathcal{X}$ of size k, the marginal weight assignments $D_{1,\mathcal{W}}$ and $D_{2,\mathcal{W}}$ (of D_1 and D_2) over $\mathcal{H}_{\mathcal{W},\mathcal{Y}}$ are identical.

2.2 Quantum Computation

A quantum algorithm is an algorithm executed on a quantum computer that produces a classical output. Most of the background in quantum computation needed to understand this paper is for the proof of Theorem 3.1. Since this proof appears in Appendix B, we present the necessary background there. In the meantime, we recall a couple basic facts about quantum computation, and refer the reader to [NC00] for a more thorough discussion.

Fact 1. Any classical computation can be implemented on a quantum computer.

Fact 2. Any function that has an efficient classical algorithm computing it can be implemented efficiently as a quantum-accessible oracle.

Fact 3. Given a quantum algorithm A with oracle access to an oracle O, each oracle O defines a probability distribution of the outputs of A. Hence, any weight assignment of oracles leads to a weight assignment of outputs of A, and if two weight assignments D_1 and D_2 are a distance ϵ apart, the weight assignment of the outputs of A under these distributions are a distance at most ϵ apart.

2.3 Cryptographic Primitives

Here we briefly outline a few cryptographic primitives, and refer to Appendix A for details. All primitives depend on a security parameter n.

An encryption scheme E is a triple of PPT algorithms $(\mathsf{E.Gen}, \mathsf{E.Enc}, \mathsf{E.Dec})$, where $\mathsf{E.Gen}(1^n)$ generates secret/public keys $(\mathsf{sk}, \mathsf{pk})$, $\mathsf{E.Enc}_{\mathsf{pk}}$ encrypts a message, and $\mathsf{E.Dec}_{\mathsf{sk}}$ decrypts a ciphertext. We use the indistinguishability under chosen plaintext attack (IND-CPA) notion of security [GM84].

An identity-based encryption (IBE) scheme $\mathsf{IBE} = (\mathsf{IBE.Gen}, \mathsf{IBE.Extract}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ is a 4-tuple of PPT algorithms where $\mathsf{IBE.Gen}(1^n)$ generates master secret/public keys $(\mathsf{msk}, \mathsf{mpk})$, $\mathsf{IBE.Extract}_{\mathsf{msk}}$ generates secret keys for given identities, $\mathsf{IBE.Enc}_{\mathsf{mpk}}$ encrypts a message to an identity, and $\mathsf{IBE.Dec}$ decrypts a ciphertext sent to an identity by using the corresponding secret key. We use the indistinguishability under chosen plaintext attack (IND-ID-CPA) notion of security [BF01].

A signature scheme $\mathsf{S} = (\mathsf{S.Gen}, \mathsf{S.Sign}, \mathsf{S.Ver})$ is a triple of PPT algorithms where $\mathsf{S.Gen}(1^n)$ generates secret/public keys $(\mathsf{sk}, \mathsf{pk})$, $\mathsf{S.Sign}_{\mathsf{sk}}$ signs a message, and $\mathsf{S.Ver}_{\mathsf{pk}}$ verifies a signature. We use the existential unforgeablility under chosen message attack (UF-CMA) notion of security [GMR88].

A pre-image sampleable function (PSF) is a quadruple of algorithms $\mathsf{F} = (\mathsf{F.Gen}, \mathsf{F.Sample}, f, f^{-1})$ where $\mathsf{F.Gen}$ generates private/public keys $(\mathsf{sk}, \mathsf{pk})$, f_{pk} is a function, $\mathsf{F.Sample}$ samples x from a distribution D such that $f_{\mathsf{pk}}(x)$ is uniform, and $f_{\mathsf{sk}}^{-1}(y)$ samples from D conditioned on $f_{\mathsf{pk}}(x) = y$. For security, we use the notion of one-wayness.

A trapdoor permutation (TDP) is a special case of a PSF where f is bijective and $\mathsf{F.Sample}$ simply returns a random element in the domain of f_{pk}. Since $\mathsf{F.Sample}$ is already defined, we omit it from the specification.

2.4 Random Oracle Model

In the Random Oracle Model, we assume the existence of a random function H, and give all parties oracle access to this function. If A makes queries to an oracle H, we denote this as A^H. If a system $\mathcal{S}$ uses a random oracle in its specification, we denote this as $\mathcal{S}^H$. The algorithms comprising any cryptographic protocol can use H, as can the adversary. Thus we modify the security games for all cryptographic systems to allow the adversary to make random oracle queries.

When a random oracle scheme is implemented, some suitable hash function H is included in the specification. Any algorithm (adversary included) now replaces oracle queries with evaluations of this hash function. In the quantum setting, because a quantum algorithm can evaluate H on an arbitrary superposition, we must allow the quantum adversary to make quantum queries to the random oracle. We call this the quantum random oracle model.

3 Distinguishing Oracles With Quantum Queries

In this section, we give some tools for arguing that quantum algorithms cannot distinguish between two distributions of oracles. The trivial way to bound the ability of any quantum algorithm to distinguish two oracle distributions D_1 and D_2 is to bound the distance between D_1 and D_2 themselves. However, in many cases, the distributions we are interested in are in fact very far apart, so we need some new techniques for arguing indistinguishability. First, we show that k-wise equivalent distributions are indistinguishable, using techniques similar to that of Meyer and Pommersheim [MP11]:

Theorem 3.1. *Let A be a quantum algorithm making q quantum queries to an oracle $H : \mathcal{X} \to \mathcal{Y}$. If we draw H from some weight assignment D, then for every z, the quantity $\Pr_{H\leftarrow D}[A^H() = z]$ is a linear combination of the quantities $\Pr_{H\leftarrow D}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$ for all possible settings of the x_i and r_i.*

This is proved in Appendix B, and immediately implies two important facts:

- If two weight assignments on oracles, D_1 and D_2, are $2q$-wise equivalent, then any q query quantum algorithm behaves the same under both weight assignments, since for all $2q$ pairs (x_i, r_i), $\Pr_{H\leftarrow D_1}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}] = \Pr_{H\leftarrow D_2}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$.
- If the quantities $\Pr_{H\leftarrow D}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$ are low-degree polynomials in some parameter λ, then so is $\Pr_{H\leftarrow D}[A^H() = z]$, since it is a linear combination of the $\Pr_{H\leftarrow D}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$. Moreover, $\Pr_{H\leftarrow D}[A^H() \in S]$ for any set S is also a low-degree polynomial.

We use the second fact to prove the following theorem:

Theorem 3.2. *Fix q, and let D_λ be a family of distributions on $\mathcal{H}_{\mathcal{X},\mathcal{Y}}$ indexed by $\lambda \in [0, 1]$. Suppose there are integers d and Δ such that for every $2q$ pairs $(x_i, r_i) \in \mathcal{X} \times \mathcal{Y}$, the function $p(\lambda) = \Pr_{H\leftarrow D_\lambda}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$ satisfies:*

- *p is a polynomial in λ of degree at most d.*
- *$p^{(i)}(0)$, the ith derivative of p at 0, is 0 for each $i \in \{1, ..., \Delta - 1\}$.*

Then any quantum algorithm A making q quantum queries can only distinguish D_λ from D_0 with probability at most $\frac{4^\Delta}{(2\Delta)!}\lambda^\Delta d^{2\Delta}$.

Proof. By the assumptions of the theorem, for any $2q$ pairs (x_i, r_i), the quantity $\Pr_{H\leftarrow D_\lambda}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$ is a polynomial of degree d in λ with the first $\Delta - 1$ derivatives at 0 being 0. By Theorem 3.1, for a q-query quantum algorithm A, $\Pr_{H\leftarrow D_\lambda}[A^H() = z]$ is a linear combination of these values. Thus, for any z, $\Pr_{H\leftarrow D_\lambda}[A^H() = z]$ is also a polynomial in λ of degree d with the first $\Delta - 1$ derivatives at 0 being 0.

Now, suppose that A distinguishes D_λ from D_0 with probability $\epsilon(\lambda)$. That is

$$\sum_z \left| \Pr_{H\leftarrow D_\lambda}[A^H() = z] - \Pr_{H\leftarrow D_0}[A^H() = z] \right| = \epsilon(\lambda) \ .$$

Fix λ_0. Let $\mathcal{Z}_{\lambda_0}$ be the set of z such that z is a more likely output under D_{λ_0} than D_0. That is, $\Pr_{H\leftarrow D_{\lambda_0}}[A^H() = z] > \Pr_{H\leftarrow D_0}[A^H() = z]$. It is not difficult to show that

$$\sum_{z\in\mathcal{Z}_{\lambda_0}} \Pr_{H\leftarrow D_{\lambda_0}}[A^H() = z] - \Pr_{H\leftarrow D_0}[A^H() = z] = \epsilon(\lambda_0)/2 \ .$$

Consider the quantity

$$p_{\lambda_0}(\lambda) \equiv \sum_{z\in\mathcal{Z}_{\lambda_0}} \Pr_{H\leftarrow D_\lambda}[A^H() = z] = \Pr_{H\leftarrow D_\lambda}[A^H() \in \mathcal{Z}_{\lambda_0}] \ .$$

Then p_{λ_0} is a degree-d polynomial such that $p^{(i)}_{\lambda_0}(0) = 0$ for $i \in \{1, ..., \Delta - 1\}$. It also lies in the range $[0, 1]$ for all $\lambda \in [0, 1]$, so we can use an inequality by the Markov brothers [DS41] to bound the Δ-th derivative for all $\lambda \in [0, 1]$:

$$\left| p^{(\Delta)}_{\lambda_0}(\lambda) \right| \leq \frac{4^\Delta \Delta!}{2(2\Delta)!} d^{2\Delta}$$

Thus we can bound $p_{\lambda_0}(\lambda) \leq p_{\lambda_0}(0) + \frac{4^\Delta}{2(2\Delta)!}\lambda^\Delta d^{2\Delta}$. Setting $\lambda_0 = \lambda$, we get

$$\epsilon(\lambda) = 2(p_\lambda(\lambda) - p_\lambda(0)) \leq \frac{4^\Delta}{(2\Delta)!}\lambda^\Delta d^{2\Delta} \ .$$

□

We conclude this section with another general approach toward arguing that no quantum algorithm can distinguish between two oracle distributions D_1 and D_2: construct a weight assignment $D_{1,2}$ which is $2q$-wise equivalent to D_1 (so that the behaviors of A under D_1 and $D_{1,2}$ are the same) and that is a distance at most ϵ from D_2 (so that the behaviors of A under $D_{1,2}$ and D_2 are ϵ-close). Then the behaviors of A under D_1 and D_2 are also ϵ-close. We formalize this latter idea with a new pseudometric on oracle distributions:

Definition 3.3. *Let D_1 and D_2 be two weight assignments on oracles. Fix k. Then define the pseudometric $|D_1 - D_2|_{(k)}$ as the minimum of $|D_{1,2} - D_2|$ over all weight assignments $D_{1,2}$ that are k-wise equivalent D_1.*

Theorem 3.4. *The minimum $D_{1,2}$ in Definition 3.3 exists, and $|D_1 - D_2|_{(k)}$ defines a pseudometric on weight assignments on oracles.*

Proof. If D is k-wise equivalent to D_1 and a distance ϵ from D_2, we say that D is a witness to $|D_1 - D_2|_{(k)} \leq \epsilon$. If $\epsilon = |D_1 - D_2|_{(k)}$, we call D an optimal witness.

First, we argue that an optimal witness exists: We see that D_1 itself is a witness to $|D_1 - D_2|_{(k)} \leq |D_1 - D_2|$. Thus to find $D_{1,2}$, we are minimizing $|D_{1,2} - D_2|$ (a continuous function in $D_{1,2}$) over the set of assignments $D_{1,2}$ that are k-wise similar to D_1 (a closed set) and at most $|D_1 - D_2|$ away from D_2 (a compact set). Thus we are minimizing over a compact set, so the minimum is actually obtained.

We now show that $|D_1 - D_2|_{(k)}$ satisfies the properties of a pseudometric:

- $|D_1 - D_1|_{(k)} = 0$, since D_1 is a witness to $|D_1 - D_1|_{(k)} \leq 0$, and $|D_1 - D_2|_{(k)}$ is non-negative for all weight assignments D_1 and D_2.
- $|D_1 - D_2|_{(k)} = |D_2 - D_1|_{(k)}$. Let $D_{1,2}$ be an optimal witness for $|D_1 - D_2|_{(k)}$. Then define $D_{2,1} = D_1 - D_{1,2} + D_2$. The sums of all weights add to 1, so this is a weight assignment. Further, it is easy to see that $|D_{2,1} - D_1| = |D_2 - D_{1,2}| = |D_1 - D_2|_{(k)}$. Further, since D_1 is k-wise equivalent to $D_{1,2}$, when looking at the marginal weight assignment over any k inputs, D_1 and $D_{1,2}$ cancel, leaving $D_{2,1}$ to be k-wise equivalent to D_2. Thus, $D_{2,1}$ is a witness to $|D_2 - D_1|_{(k)} \leq |D_1 - D_2|_{(k)}$. We can perform the same argument in reverse to get the other inequality.
- $|D_1 - D_3|_{(k)} \leq |D_1 - D_2|_{(k)} + |D_2 - D_3|_{(k)}$. Let $D_{1,2}$ and $D_{2,3}$ be the optimal witnesses for $|D_1 - D_2|_{(k)}$ and $|D_2 - D_3|_{(k)}$. Define $D_{1,3} = D_{1,2} - D_2 + D_{2,3}$. It is easy to see that D_3 is a weight assignment and that $|D_{1,3} - D_3| \leq |D_{1,2} - D_2| + |D_{2,3} - D_3|$. It is also not difficult to show that, since $D_{2,3}$ is k-wise equivalent to D_2, $D_{1,3}$ must be k-wise equivalent to $D_{1,2}$, which is k-wise equivalent to D_1. Hence $D_{1,3}$ is a witness for the identity in question. □

Observe that $|D_1 - D_2|_{(k)} = 0$ if and only if D_1 is k-wise equivalent to D_2. Thus, we can rephrase Theorem 3.1 as follows: if $|D_1 - D_2|_{(2q)} = 0$, then no quantum algorithm making q quantum queries can distinguish the oracle weight assignments D_1 and D_2. We now generalize this to $|D_1 - D_2|_{(2q)} \neq 0$:

Corollary 3.5. *Let D_1 and D_2 be two oracle distributions. Then the output weight assignment of any q-query quantum algorithm A under D_1 and D_2 are $|D_1 - D_2|_{(2q)}$-close.*

Proof. Let $D_{1,2}$ be an optimal witness for $|D_1 - D_2|_{(2q)}$. $D_{1,2}$ is $2q$-wise equivalent to D_1, so by Theorem 3.1, the behaviors of the A under D_1 and $D_{1,2}$ are identical. The behavior of A under D_2 is $|D_{1,2} - D_2| = |D_1 - D_2|_{(2q)}$-close to that under $D_{1,2}$, and hence D_1. □

We note that a slightly weaker version of Theorem 3.2 can also be proved using the above ideas. Roughly, we pick some constants λ_i and find functions $a_i(\lambda)$ such that

$$p(\lambda) = \left(1 - \sum_{i=0}^{d-\Delta} a_i(\lambda)\right) p(0) + \sum_{i=0}^{d-\Delta} a_i(\lambda) p(\lambda_i)$$

for any polynomial p of degree d such that $p^{(i)}(0) = 0$ for $i \in \{1, ..., \Delta - 1\}$. The $a_i(\lambda)$ are basically modified Lagrange interpolating polynomials. Next, we define a new weight assignment

$$D'_\lambda = \left(1 - \sum_{i=0}^{d-\Delta} a_i(\lambda)\right) D_0 + \sum_{i=0}^{d-\Delta} a_i(\lambda) D_{\lambda_i} \ .$$

By defining D'_λ in this way, D'_λ turns out to be $2q$-wise equivalent to D_λ. Because D_{λ_i} are all distributions, the distance between D'_λ and D_0 is at most $2\sum_{i=0}^{d-\Delta} |a_i(\lambda)|$, thus showing that $|D_\lambda - D_0|_{(2q)}$ also satisfies this bound. By picking the right values for λ_i, we can get this bound to be small. We can then use Corollary 3.5 to bound the distance between the output distributions of the q-query quantum algorithm A.

4 Semi-Constant Distributions

In this section, we define a class of distributions on oracles in $\mathcal{H}_{\mathcal{X},\mathcal{Y}}$ called semi-constant distributions. Our motivation for these distributions is to support quantum Full Domain Hash-type arguments, where a random value is inserted into a small but significant fraction of oracle inputs. The following definition captures the essence of the this idea:

Definition 4.1. *Define SC_λ, the semi-constant distribution, as the distribution over $\mathcal{H}_{\mathcal{X},\mathcal{Y}}$ resulting from the following process:*

- *First, pick a random element y from $\mathcal{Y}$.*
- *For each $x \in \mathcal{X}$, do one of the following:*
  - *With probability λ, set $H(x) = y$. We call x a distinguished input to H.*
  - *Otherwise, set $H(x)$ to be a random element in $\mathcal{Y}$.*

First, notice that SC_0 is the uniform distribution, since in this case, the probability that an input is distinguished is 0. We bound the ability of a quantum algorithm to distinguish between SC_λ and $\mathsf{SC}_0 = U$ using the ideas of Section 3.

Lemma 4.2. *Fix k. For each k pairs (x_i, r_i), the probability $\Pr_{H \leftarrow \mathsf{SC}_\lambda}[H(x_i) = r_i \forall i \in \{1, ..., k\}]$ is a degree-k polynomial in λ whose first derivative is 0 at $\lambda = 0$.*

This, proved in Appendix C, shows that for any q_H, we can set $d = k = 2q_H$ and $\Delta = 2$ in the assumptions of Theorem 3.2. We immediately get:

Corollary 4.3. *The distribution of outputs of a quantum algorithm making q_H queries to an oracle drawn from SC_λ is at most a distance $\frac{8}{3}q_H^4\lambda^2$ away from the case when the oracle is drawn from the uniform distribution.*

We note that using standard quantum query results, it is possible to prove this corollary for a bound that is first-order in λ. However, we will see in Section 5 that we need a bound that is second-order in λ for our security results.

We do not know if this bound is tight. In Appendix C, we adapt the collision search algorithm of Brassard et al. [BHT97] to SC_λ, obtaining a distinguishing probability of $\Omega(q_H^3\lambda^2)$.

5 Quantum Security Arguments

In this section, we explore the random oracle proof technique of Bellare and Rogaway [BR93]. In this technique, we plug a challenge c into a randomly chosen hash query, and hope that the adversary uses c to break the system. If so, we can use the adversary to solve the underlying problem.

In the quantum setting, the straight-forward application of this technique breaks down. The adversary can now query the hash function on a superposition of exponentially many inputs. If we plug c to one of those inputs, the probability that the adversary uses c is exponentially small.

We instead plug c into some small but significant fraction of the inputs, so that the oracle is distributed according to SC_λ. As we have shown, a quantum algorithm cannot detect that this oracle is not random, but now the adversary uses c with significant probability.

5.1 Identity-Based Encryption from Lattices

Here we prove the security of the IBE scheme from Gentry et al. [GPV08]. Their scheme is constructed from an encryption scheme $\mathsf{E} = (\mathsf{E.Gen}, \mathsf{E.Enc}, \mathsf{E.Dec})$, for which there exists a trapdoor allowing the computation of secret keys from public keys.

More specifically, let $\mathsf{F} = (\mathsf{F.Gen}, \mathsf{F.Sample}, f, f^{-1})$ be a pre-image sampleable function (PSF). Suppose $\mathsf{E.Gen}(1^n)$ works as follows: generate $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{F.Gen}(1^n)$. Then, sample $\mathsf{sk} \leftarrow \mathsf{F.Sample}(1^n)$, and compute $\mathsf{pk} = f_{\mathsf{mpk}}(\mathsf{sk})$. Output $(\mathsf{sk}, (\mathsf{pk}, \mathsf{mpk}))$.

Gentry et al. give such an encryption scheme based on the hardness of lattice problems. Their security reduction treats the adversary as a black box, so their proof holds in the quantum setting. They then prove the security of the IBE scheme $\mathsf{IBE} = (\mathsf{IBE.Gen} = \mathsf{F.Gen}, \mathsf{IBE.Extract}^H, \mathsf{IBE.Enc}^H, \mathsf{IBE.Dec})$ where H maps identities to public keys of E and:

- $\mathsf{IBE.Extract}_{\mathsf{msk}}(\mathsf{id})^H = f^{-1}_{\mathsf{msk}}(H(\mathsf{id}))$
- $\mathsf{IBE.Enc}_{\mathsf{mpk}}(\mathsf{id}, m)^H = \mathsf{E.Enc}_{H(\mathsf{id})}(m)$
- $\mathsf{IBE.Dec}_{\mathsf{sk}_{\mathsf{id}}}(c) = \mathsf{E.Dec}_{\mathsf{sk}_{\mathsf{id}}}(c)$

Theorem 5.1. *Let E and F be as above, and suppose that E is quantum IND-CPA-secure. If we model H as a random oracle, then IBE is quantum IND-ID-CPA-secure.*

Proof. Let A be a quantum adversary making q_H hash queries, q_E extract queries queries, that breaks IBE with advantage ϵ.

Let Game_0 be the standard attack game for IBE: the challenger generates $(\mathsf{msk}, \mathsf{mpk})$ from $\mathsf{IBE.Gen}$, and sends mpk to the adversary. The adversary can make (classical) extraction queries on identities id_i, to which the challenger responds with $\mathsf{IBE.Extract}^H_{\mathsf{msk}}(\mathsf{id}_i)$, and (quantum) hash queries to the random oracle H. A then produces an identity id^*, along with messages m_0 and m_1. The challenger chooses a random bit b, and responds with $\mathsf{IBE.Enc}_{\mathsf{mpk}}(\mathsf{id}, m_b)$. A is allowed to make more extraction and hash queries, and produces a bit b'. A wins if $b' = b$ and for all i, $\mathsf{id}_i \neq \mathsf{id}^*$. By definition, this happens with probability $\frac{1}{2} + \epsilon$.

Assume without loss of generality that A never asks for the secret key corresponding to id^*. Let A' be the following algorithm that makes quantum queries to an oracle H, and simulates the interaction between A and the challenger: generate $(\mathsf{msk}, \mathsf{mpk})$ from $\mathsf{IBE.Gen}$, send mpk to A. Run A, forwarding all hash queries to H, and answering extraction queries using $\mathsf{IBE.Extract}^H$. Similarly, answer the challenge query by choosing a random b and encrypting m_b to the identity id generated by A. The output of A' is (c, id^*, Q) where $c = b \oplus b'$ and Q is the list of extraction queries made by A. We can now think of Game_0 as follows: run A' with a random oracle H to obtain (c, id^*, Q). Report that the game is won if and only if $c = 0$. The number of queries to H made by A' is q_H for direct queries, q_E for queries through the extract algorithm, and 1 for the encryption of m_b, for a total of $q_H + q_E + 1$ queries.

Let $\lambda \in (0, 1)$ to be chosen later. Let $\mathcal{M}$ be the identity space. Define Game_1 as follows: first, pick a subset $\mathcal{X}$ of $\mathcal{M}$ where each $\mathsf{id} \in \mathcal{M}$ is put in $\mathcal{X}$ with independent probability λ. Run A' as before to obtain (c, id^*, Q). However, now we decide if the game is won as follows: if $\mathsf{id}^* \in \mathcal{X}$ and $Q \bigcap \mathcal{X} = \emptyset$, then the game is won if $c = 0$. Otherwise, flip a random coin to decide if the game is won. By our assumption that A never asks for a secret key for the challenge identity, we must have that $\mathsf{id}^* \notin Q$. Therefore, every identity in Q, as well as id^*, has an independent probability λ of being in $\mathcal{X}$. Thus, the conditions to not flip a coin are met with probability at least $\lambda(1 - \lambda q_E)$, independent of the probability that $c = 0$. Thus, Game_1 is won with probability at least $\frac{1}{2} + \lambda(1 - \lambda q_E)\epsilon \geq \frac{1}{2} + \lambda\epsilon - q_E\lambda^2$.

Let Game_2 be Game_1, except that we choose a random pk in the public key space of E, and set $H(x) = pk$ for all x in $\mathcal{X}$, and choose $H(x)$ randomly for all other x. H is now distributed according to SC_λ.

Claim 1. *There exists a polynomial $\ell(\cdot, \cdot)$ such that A' wins Game_2 with probability at least*

$$\frac{1}{2} + \lambda\epsilon - \frac{\ell(q_H, q_E)}{4}\lambda^2 \quad .$$

We defer the precise proof to Appendix D, and instead give an informal justification: by Corollary 4.3, the output distribution of A' in Game_2 is at most a distance $\frac{8}{3}(q_H + q_E + 1)^4\lambda^2$ from that of Game_1. Thus, we would expect A' to win Game_2 with probability at least $\frac{1}{2} + \lambda\epsilon - \left(q_E + \frac{8}{3}(q_H + q_E + 1)^4\right)\lambda^2$, which has the desired form. Of course, deciding if A' wins Game_2 depends on the set of distinguished inputs $\mathcal{X}$, so a more careful analysis is required to justify the claim.

Let us now restate Game_2 in terms of A: pick $\mathcal{X}$ and H as above, and perform the standard attack game for IBE using H as the random oracle. When A produces b', first check that no extraction queries were in $\mathcal{X}$ and that the identity id^* produced by A *is* in $\mathcal{X}$. If so, A wins if and only if $b = b'$. Otherwise, if the check fails, flip a coin to decide if A wins. We can now make the following observation: once an extraction query in $\mathcal{X}$ is made or the challenge query with $\mathsf{id}^* \notin \mathcal{X}$ is made, we can abort running A, since we no longer need A to determine if the game is won.

Now we describe an adversary B that breaks E. Assume B has quantum access to two random oracles O_1 and O_2. O_1 maps identities to randomness used by Sample. O_2 maps identities to bits, outputting 1 with probability is λ. In Section 6, we will show how to efficiently simulate these oracles. On input $(\mathsf{mpk}, \mathsf{pk})$, B works as follows:

- Send mpk to A and simulate A, playing the role of challenger to A.
- Construct a (quantum) oracle H such that

$$H(\mathsf{id}) = \begin{cases} \mathsf{pk} & \text{if } O_2(\mathsf{id}) = 1 \\ f_{\mathsf{mpk}}(\mathsf{Sample}(; O_1(\mathsf{id}))) & \text{otherwise} \end{cases}$$

  where $\mathsf{Sample}(; r)$ means run Sample with randomness r.
- When A asks for the secret key for id_i, compute $O_2(\mathsf{id}_i)$. If the result is 1, output a random bit and abort. Otherwise, respond with $\mathsf{sk}_i = \mathsf{Sample}(; O_1(\mathsf{id}_i))$.
- When A produces the challenge query $(\mathsf{id}^*, m_0, m_1)$, check if $O_2(\mathsf{id}^*) = 1$. If so, send (m_0, m_1) to B's challenger. Otherwise, output a random bit and abort.
- When the challenger responds with a ciphertext, send it to A.
- When A outputs a bit b', output the same bit.

Let $\mathcal{X}$ be the set of identities id such that $O_2(\mathsf{id}) = 1$. We can then see that the abort conditions are equivalent to Game_2. For each extract query id_i that succeeds, we have that $O_2(\mathsf{id}_i) = 0$, so

$$f_{\mathsf{mpk}}(\mathsf{sk}_i) = f_{\mathsf{mpk}}(\mathsf{Sample}(; O_1(\mathsf{id}_i))) = H(\mathsf{id}_i)$$

Thus sk_i is a correct secret key for id_i, and it is distributed correctly (since it is a random pre-image of $H(\mathsf{id}_i)$). If B does not abort on the challenge query, it means $O_2(\mathsf{id}^*) = 1$, so $H(\mathsf{id}^*) = \mathsf{pk}$. Also, we can see that H is distributed according to SC_λ. Therefore, if B aborts, we win with probability $\frac{1}{2}$, and if we do not abort, we win if and only if A wins. Thus, the view of A as a subroutine of B is the same as in Game_2, and B wins with the same probability that A does. Therefore, B wins with probability at least $\frac{1}{2} + \lambda\epsilon - \frac{\ell(q_H, q_E)}{4}\lambda^2$.

The advantage of B is at least $\lambda\epsilon - \ell(q_H, q_E)\lambda^2/4$. We can now choose λ: the maximum advantage occurs when $\lambda = 2\epsilon/\ell(q_H, q_E)$, which gives

$$Adv_B \geq \frac{\epsilon^2}{\ell(q_H, q_E)}$$

Thus, if A had non-negligible advantage, so does B. □

5.2 Hierarchical IBE from Lattices

In this section, we show the general idea behind adapting the techniques above to proving the security of the hierarchical identity-based encryption (HIBE) schemes of Agrawal et al. [ABB10] and Cash et al. [CHKP10].

In a HIBE scheme, identities are structured as a tree, with the identity of any node containing the identity of its parent as a proper prefix. Any node on the tree can produce private keys for any nodes in the subtree rooted at that node. We allow an adversary to adaptively take control of an arbitrary number of nodes in the tree (and thus the subtrees rooted there). An HIBE scheme

is secure if the adversary cannot decrypt messages encrypted to an identity id^* of the adversaries choice but not under its control.

In [ABB10], the random oracle scheme has an oracle H that maps identities to some random quantities. The reduction has the following high-level structure:

- Guess which level w of the tree contains the identity id^*
- For each level i, generate some random quantities R_i.
- For each level i, simulate a separate random oracle for identities at that level. For i, guess which query number q_i will contain the hash of the level-i parent of id^*.
- Answer the jth random oracle query at level i on $\mathsf{id}_{i,j}$ as follows: if $j = q_i$, output R_i. Otherwise, output a random value.
- Answer secret key queries on id in some special way, but fail if for all prefixes id_i of id, $\mathsf{id}_i = \mathsf{id}_{i,q_i}$. That is, fail if $H(\mathsf{id}_i) = R_i$.
- When the adversary generates the identity id^*, we succeed if both the adversary succeeds and if id^* is at level w and all prefixes id_i^* of id^* satisfy $\mathsf{id}_i^* = \mathsf{id}_{i,q_i}$.

We now show how to prove security by repeatedly applying the arguments of Theorem 5.1. Basically, we iterate over levels i, and insert R_i into a λ_i fraction of identities at that level. In iteration i, we say the adversary wins if it won in the previous iteration, the level-i prefix of the chosen identity id^* is in the λ_i fraction of distinguished identities (that is, $H(\mathsf{id}_i^*) = R_i$), and no signature query is. Let $\ell = \ell(q_H, q_E)$, where $\ell(\cdot,\cdot)$ is the polynomial from Theorem 5.1. If the iteration i advantage is ϵ_i, then using the same techniques as in Theorem 5.1, we can set λ_i so that

$$\epsilon_i \geq \frac{\epsilon_{i-1}^2}{\ell}$$

We will say that in iteration 0, the adversary wins if it normally would win and we guessed which level id^* belonged to correctly. That is, $\epsilon_0 = \epsilon/d$, where ϵ is the adversary's advantage in the standard game. This gives us a total advantage after iteration d of at least

$$\frac{(\epsilon/d)^{2^d}}{\ell^{(2^d-1)}} = \ell\left(\frac{\epsilon}{d\ell}\right)^{2^d}$$

Notice that the dependence on d in doubly-exponential, whereas in the classical proof it was singly exponential. Thus, for the same security parameters, this proof only works for much smaller depth than the classical proof.

These techniques apply as well to the random oracle HIBE of Cash et al. [CHKP10], though their reduction is a bit more complicated, as there is a second random oracle G which needs to be handled in a similar way.

5.3 Signatures from Trapdoor Permutations

Here we discuss the security of the Full Domain Hash (FDH) signature scheme:

Definition 5.2 (FDH Signatures)**.** *Let* $\mathsf{F} = (\mathsf{F.Gen}, f, f^{-1})$ *be a trapdoor permutation, and a hash function* H *that maps messages to images of* f*, let* $\mathsf{S}^H = (\mathsf{S.Gen} = \mathsf{F.Gen}, \mathsf{S.Sign}^H, \mathsf{S.Ver}^H)$ *where:*

- $\mathsf{S.Sign}_{\mathsf{sk}}^H(m) = f_{\mathsf{sk}}^{-1}(H(m))$

- $\mathsf{S.Ver}^H_{\mathsf{pk}}(m,\sigma) = \begin{cases} \mathsf{accept} & \textit{if } f_{\mathsf{pk}}(\sigma) = H(m) \\ \mathsf{reject} & \textit{ortherwise} \end{cases}$

We now state the main theorem of this section:

Theorem 5.3. *Let* F *be a quantum one-way trapdoor permutation. If we model* H *as a random oracle, then* S *is quantum UF-CMA-secure*

The proof of this theorem is very similar to that of Theorem 5.1, and appears in Appendix E.

6 Simulating Random Oracles

In this last section, we explain how to efficiently simulate random oracles. All quantum random oracle proofs to date require the reduction algorithm to have quantum access to a random oracle, but a truly random oracle requires exponentially many bits of randomness to construct. We show that this is not a problem:

Theorem 6.1. *Any quantum algorithm* A *making quantum queries to random oracles* O_i *can be efficiently simulated by a quantum algorithm* B*, which has the same output distribution, but makes no queries.*

Proof. We construct an algorithm B which simulates A, and answers queries to oracle O_i with evaluations of efficient functions f_i. Boneh et al. [BDF+11] use pseudorandom functions (PRF) for the f_i. At first glance, this seems like the only option, as we need a function f_i that cannot be distinguished from random.

Notice, however, that f_i need not be secure against all adversaries, just the adversary we are simulating. We know that our adversary makes q_i queries to oracle O_i, so it suffices to have f_i be PRFs secure for up to q_i quantum queries. In the classical setting, q_i-wise independent functions (functions that are q_i-wise equivalent to a random function) serve as *perfectly* secure PRFs for up to q_i classical queries. We could hope that something similar holds in the quantum world: indeed, according to Theorem 3.1, if f_i is $2q_i$-wise equivalent to a random function, then the behavior of our adversary is the same when the oracle is random and when it is f_i. Thus if the f_i are $2q_i$-wise independent, algorithm A, as a subroutine of B, behaves identically to the case where A is given truly random oracles. Hence, the output distribution of B is identical to that of A.

Efficient constructions of k-wise independent functions have been known for some time [Jof74, KM94], and they have been used extensively in the derandomization literature [Lub85, ABI86, KM93]. One common approach to construct a k-wise independent function f from $\mathcal{X}$ to $\mathcal{Y}$ is to assume that $N = |\mathcal{Y}|$ is a prime power and interpret $\mathcal{Y}$ as the field $\mathbb{F}_N$. Then define a matrix C with the following properties:

- The entries are elements in $\mathbb{F}_N$.
- There are $|\mathcal{X}|$ rows and some small number r of columns.
- Each subset of k rows is linearly independent.

One such example is the Vandermonde matrix, which is used by Alon et al.[ABI86]. To define the function f, we then pick a random vector v in $\mathbb{F}_N^r$. $f(x)$ is then the xth element of of the vector Cv. Since any k rows of C are linearly independent, any k elements of Cv are independent, and hence f is k-wise independent. The key to making this efficient is that to compute $f(x)$, we only need the xth row of C, which we can compute on the fly. □

Hence, we can simulate O_1 from Theorem 5.1. To simulate O_2, which outputs a bit such that $O_2(x) = 1$ with probability λ, approximate λ by some rational number a/b where b is a prime power, and construct a k-wise independent function f' with range $\mathcal{Y} = \{1, ..., b\}$. Then set

$$f(x) = \begin{cases} 1 & \text{if } f'(x) \leq a \\ 0 & \text{otherwise} \end{cases}$$

7 Conclusion

We have shown how to adapt certain classical random oracle arguments to the quantum random oracle model. Specifically, we gave quantum security proofs for the IBE scheme of Gentry et al. [GPV08] and the Full Domain Hash signature scheme. We achieved this by defining a distribution of oracles, called semi-constant distributions, and showing that such oracles cannot be distinguished from a random oracle by a quantum adversary. We also show how these techniques can be applied to the random oracle HIBE schemes of Cash et al. [CHKP10] and Agrawal et al. [ABB10]. Lastly, we have shown how to remove the need for quantum-secure pseudorandom functions from prior work.

Although we have made progress toward converting classical random oracle proofs into quantum proofs, there is still much work to be done. First, reductions using our technique result in an algorithm whose advantage is the square of the advantage of the underlying adversary, divided by some other factors. While improving the reduction to first-order in the adversary's advantage would still be of interest for the IBE scheme and Full Domain Hash, it would not affect the qualitative security statements. However, for the HIBE schemes, this second-order behavior results in an exponential weakening of the security relative to the classical reduction for deep identity trees. Therefore, to get the same qualitative security statements as in the classical world, our technique needs to be refined to make the reduction first-order in ϵ.

Further work lies in proving the security of other Random Oracle Model schemes. For example, the construction of signatures from identification protocols by Fiat and Shamir [FS87], though similar to the proofs in this paper, still needs a quantum proof. The difficulty in the Fiat-Shamir reduction is that the plugging step initiates the underlying identification protocol, and there is no obvious quantum analog for this strategy. Also, different types of security arguments, such as Fujisaki and Okamoto's generic conversion of weakly secure encryption schemes into a CCA-secure encryption scheme [FO99], still lack a quantum proof of security.

Acknowledgments

We would like to thank Dan Boneh for his guidance and many insightful discussions. This work was supported by NSF and DARPA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

References

[Aar09] Scott Aaronson. Quantum Copy-Protection and Quantum Money. *Proceedings of the 24th Annual IEEE Conference on Computaitonal Complexity (CCC)*, 2009.

[ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. *Advances in Cryptology - CRYPTO 2010*, pages 98–115, 2010.

[ABI86] Noga Alon, László Babai, and Alon Itai. A Fast and Simple Randomized Parallel Algorithm for the Maximal Independent Set Problem. *Journal of algorithms*, 7(4):567–583, 1986.

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and Weaknesses of Quantum Computing. *SIAM Journal on Computing*, 26:1510–1523, 1997.

[BDF+11] Dan Boneh, Ozgur Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random Oracles in a Quantum World. In *Advances in Cryptology - ASIACRYPT 2011*, 2011.

[BF01] Dan Boneh and Matthew Franklin. Identity-Based Encryption from the Weil Pairing. *Advances in Cryptology - CRYPTO 2001*, 32(3):586–615, 2001.

[BF11] Dan Boneh and David Freeman. Homomorphic Signatures for Polynomial Functions. *Advances in Cryptology - EUROCRYPT 2011*, 6632:149–168, 2011.

[BHK+11] Gilles Brassard, Peter Høyer, Kassem Kalach, Marc Kaplan, Sophie Laplante, and Louis Salvail. Merkle Puzzles in a Quantum World. *Advances in Cryptology - CRYPTO 2011*, pages 391–410, 2011.

[BHT97] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum Algorithm for the Collision Problem. *ACM SIGACT News (Cryptology Column)*, 28:14–19, 1997.

[BR93] Mihir Bellare and Philip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS)*, number November 1993, pages 62–73. ACM, 1993.

[BS08] Gilles Brassard and Louis Salvail. Quantum Merkle Puzzles. *Second International Conference on Quantum, Nano and Micro Technologies (ICQNM 2008)*, pages 76–79, February 2008.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai Trees, or How to Delegate a Lattice Basis. *Advances in Cryptology - EUROCRYPT 2010*, pages 523–552, 2010.

[DS41] R J Duffin and A C Schaffer. A Refinement of an Inequality of the Brothers Markoff. *Trans. Amer. Math. Soc*, 44(3):289–297, 1941.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Advances in Cryptology - CRYPTO 1999*, pages 79–79. Springer, 1999.

[FS87] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology - CRYPTO 1986*, pages 186–194. Springer, 1987.

[GKV10] S. Dov Gordon, Jonathan Katz, and Vinod Vaikuntanathan. A Group Signature Scheme from Lattice Assumptions. *Advances in Cryptology - ASIACRYPT 2010*, pages 395–412, 2010.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of computer and system sciences*, pages 270–299, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for Hard Lattices and New Cryptographic Constructions. *Proceedings of the 40th Annual ACM symposium on Theory of computing (STOC)*, page 197, 2008.

[Gro96] Lov K. Grover. A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*, pages 212–219, 1996.

[HSS11] Sean Hallgren, Adam Smith, and Fang Song. Classical Cryptographic Protocols in a Quantum World. *Advances in Cryptology - CRYPTO 2011*, pages 411–428, 2011.

[Jof74] A. Joffe. On a Set of Almost Deterministic k-Independent Random Variables. *The Annals of Probability*, 2(1):161–162, 1974.

[KM93] Daphne Koller and Nimrod Megiddo. Constructing Small Sample Spaces Satisfying Given Constraints. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC)*, pages 268–277. ACM, 1993.

[KM94] Howard Karloff and Yishay Mansour. On Construction of k -Wise Independent Random Variables. *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC)*, 17:564–573, 1994.

[Lub85] Michael Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. In *Proceedings of the 17th Annual ACM symposium on Theory of computing (STOC)*, pages 1–10. ACM, 1985.

[MP11] David Meyer and James Pommersheim. On the Uselessness of Quantum Queries. *Theoretical Computer Science*, (March):1–12, 2011.

[NC00] Michael A. Nielsen and Isaac Chuang. Quantum Computation and Quantum Information. *American Journal of Physics*, 70(5):558, 2000.

[Sho97] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Unr10] Dominique Unruh. Universally Composable Quantum Multi-Party Computation. *Advances in Cryptology - EUROCRYPT 2010*, pages 486–505, 2010.

[Zha12] Mark Zhandry. How to Construct Quantum Random Functions, April 2012. Full version available at the Cryptology ePrint Archives: http://eprint.iacr.org/2012/182/.

A Cryptographic Primitives

Definition A.1 (Encryption Scheme)**.** *A public key encryption scheme* E *is a triple of PPT algorithms* $(\mathsf{E.Gen}, \mathsf{E.Enc}, \mathsf{E.Dec})$ *where*

- $\mathsf{E.Gen}(1^n)$ *generates a secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$*.*
- $\mathsf{E.Enc}_{\mathsf{pk}}(m)$ *computes a ciphertext* c*.*
- $\mathsf{E.Dec}_{\mathsf{sk}}(c)$ *computes the plaintext* m*, such that* $\mathsf{E.Dec}_{\mathsf{sk}}(\mathsf{E.Enc}_{\mathsf{pk}}(m)) = m$*.*

We use the standard chosen plaintest attack (CPA) game [GM84] to model security:

- The challenger generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{E.Gen}(1^n)$, and sends pk to the adversary A.
- A generates the challenge plaintext (m_0, m_1).
- The challenger chooses a bit b, and responds with the challenge ciphertext $c = \mathsf{E.Enc}_{\mathsf{pk}}(m_b)$.
- A makes a guess b' for b.

The advantage of A is ϵ, where $\frac{1}{2} + \epsilon$ is the probability that $b = b'$. We say that E is (quantum) IND-CPA-secure if all efficient (quantum) algorithms have only negligible advantage (in n).

An identity-based encryption (IBE) scheme IBE is an encryption scheme where identities serve as public keys, and a trusted authority hands out the corresponding secret keys. In particular:

Definition A.2 (IBE Scheme)**.** *An identity-based encryption (IBE) scheme* IBE *is a quadruple of PPT algorithms* $(\mathsf{IBE.Gen}, \mathsf{IBE.Extract}, \mathsf{IBE.Enc}, \mathsf{IBE.Dec})$ *where*

- $\mathsf{IBE.Gen}(1^n)$ *generates a master secret/public key pair* $(\mathsf{msk}, \mathsf{mpk})$*.*
- *The trusted authority uses* $\mathsf{IBE.Extract}_{\mathsf{msk}}(\mathsf{id})$ *to compute a secret key* $\mathsf{sk}_{\mathsf{id}}$ *corresponding to the identity* id*.*
- $\mathsf{IBE.Enc}_{\mathsf{mpk}}(\mathsf{id}, m)$ *encrypts* m *to identity* id*.*
- $\mathsf{IBE.Dec}_{\mathsf{sk}_{\mathsf{id}}}(c)$ *decrypts* c*, such that*

$$\mathsf{IBE.Dec}_{\mathsf{IBE.Extract}_{\mathsf{msk}}(\mathsf{id})}(\mathsf{IBE.Enc}_{\mathsf{mpk}}(\mathsf{id}, m)) = m$$

We use the standard CPA attack game for IBE schemes, where the adversary can choose the identity id for which it will attempt to decrypt, and can get the secret keys corresponding to any other identities [BF01]. Specifically,

- The challenger generates $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{IBE.Gen}(1^n)$, and sends mpk to the adversary A.
- The adversary can make extraction queries on identities id_i, to which the challenger responds with $\mathsf{IBE.Extract}_{\mathsf{msk}}(\mathsf{id}_i)$.
- The adversary generates an identity id, and two messages (m_0, m_1).
- The challenger chooses a random bit b, and responds with $\mathsf{IBE.Enc}_{\mathsf{id}}(m_b)$.
- A makes a guess b' for b.

The advantage of A is ϵ, where $\frac{1}{2} + \epsilon$ is the probability that $\mathsf{id} \neq \mathsf{id}_i$ for any i and $b = b'$. We say that IBE is (quantum) IND-ID-CPA-secure if all efficient (quantum) algorithms have only negligible advantage (in n).

Definition A.3 (Signature Scheme)**.** *A signature scheme* S *is a triple of PPT algorithms* $(\mathsf{S.Gen}, \mathsf{S.Sign}, \mathsf{S.Ver})$ *where*

- $\mathsf{S.Gen}(1^n)$ *generates a secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$.
- $\mathsf{S.Sign}_{\mathsf{sk}}(m)$ *computes a signature* σ *on the message* m.
- $\mathsf{S.Ver}_{\mathsf{pk}}(m, \sigma)$ *outputs* accept *or* reject*, such that* $\mathsf{S.Ver}_{\mathsf{pk}}(m, \mathsf{S.Sign}_{\mathsf{sk}}(m)) = \mathsf{accept}$.

We will use the standard chosen message attack (CMA) game to define security [GMR88]:

- The challenger generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{S.Gen}(1^n)$, and sends pk to the adversary A.
- A can make signature queries on messages m_i, to which the challenger responds with $\mathsf{S.Sign}_{\mathsf{sk}}(m_i)$.
- A produces a forgery candidate (m, σ).

The advantage of A is the probability that $m \neq m_i$ for any i and $\mathsf{Ver}_{\mathsf{pk}}(m, \sigma) = \mathsf{accept}$. We say that S is (quantum) UF-CMA-secure if all efficient (quantum) adversaries have negligible advantage (in n).

Definition A.4 (PSF)**.** *A pre-image sampleable function (PSF) is a quadruple of algorithms* $\mathsf{F} = (\mathsf{F.Gen}, \mathsf{F.Sample}, f, f^{-1})$ *where:*

- $\mathsf{F.Gen}(1^n)$ *generates a secret/public key pair* $(\mathsf{sk}, \mathsf{pk})$.
- f_{pk} *is a function.*
- $\mathsf{F.Sample}$ *samples* x *from a distribution* D *such that* $f_{\mathsf{pk}}(x)$ *is uniform.*
- $f_{\mathsf{sk}}^{-1}(y)$ *samples from* D *conditioned on* $f_{\mathsf{pk}}(x) = y$.

We model security with the following game:

- The challenger generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{F.Gen}(1^n)$, and sends pk to the adversary A. It also sends $y = f_{\mathsf{pk}}(\mathsf{F.Sample}())$ to A.
- A makes a guess x.

The advantage of A is the probability that $f_{\mathsf{pk}}(x) = y$. We say that F is (quantum) one-way if all efficient (quantum) algorithms have negligible advantage.

A trapdoor permutation (TDP) is a special case of a PSF where f is bijective and $\mathsf{F.Sample}$ simply returns a random element in the domain of f_{pk}. Since $\mathsf{F.Sample}$ is already defined, we omit it from the specification.

B Quantum Computation

We give some background on quantum computation. For a more thorough discussion, please see [NC00]. We also prove Theorem 3.1, which states that the output distribution of a quantum algorithm making q_H queries to a quantum oracle drawn from some distribution D depends only on the marginal distributions of D on all subsets of $2q$ inputs.

Let $\mathcal{H}$ be a Hilbert space with an inner product $\langle \cdot | \cdot \rangle$, and let $B = \{|x\rangle\}$ be an orthonormal basis for $\mathcal{H}$, index by a parameter $x \in \mathcal{X}$. The state of a quantum system on $\mathcal{H}$ is specified by a vector $|\phi\rangle \in \mathcal{H}$ of norm 1.

Quantum Measurement Given a state $|\phi\rangle$, we can measure $|\phi\rangle$ in the basis B, obtaining the value $x \in \mathcal{X}$ with probability $|\langle x|\phi\rangle|^2$. Thus, to each $|\phi\rangle$, we associate a distribution D_ϕ where $D_\phi(x) = |\langle x|\phi\rangle|^2$. The normalization constant and the fact that B is an orthonormal basis ensure that this is in fact a valid distribution. After measurement, the system is in state $|x\rangle$.

If $B = \{|x, y\rangle\}$ for $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, then we can also perform a partial measurement over $\mathcal{X}$. The distribution over $\mathcal{X}$ is the marginal distribution of D_ϕ when restricted to $\mathcal{X}$. That is, we obtain the value x with probability $\sum_{y\in\mathcal{Y}} |\langle x, y|\phi\rangle|^2$. The resulting state is the result of projecting $|\phi\rangle$ to the space spanned by $\{|x, y\rangle\}$ for $y \in \mathcal{Y}$, and renormalizing so that the norm is 1.

Quantum Algorithms A quantum algorithm A over a Hilbert space $\mathcal{H}$ with an orthonormal basis B is specified by unitary transformation U. The input to A is an element $x_0 \in X$. The system is initialized to the basis state $|x_0\rangle$, and U is applied to the system, obtaining the final state $|\phi\rangle = U|x_0\rangle$. Then the state is sampled according to the distribution D_ϕ.

Let $\mathcal{X}$ and $\mathcal{Y}$ be sets such that $\mathcal{Y}$ is a commutative group with addition operation $\oplus$. For notational convenience, we assume that every element has order at most 2 ($y \oplus y$ is the identity for all $y \in \mathcal{Y}$). Given a function $H : \mathcal{X} \to \mathcal{Y}$ and third set $\mathcal{Z}$, define the orthonormal basis B as the set $\{|x, y, z\rangle\}$ for $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and $z \in \mathcal{Z}$. Define the unitary transformation $H_{\texttt{trans}}$ over the Hilbert space spanned by B as the transformation that takes $|x, y, z\rangle$ into $|x, y \oplus H(x), z\rangle$. $H_{\texttt{trans}}$ is unitary, it's own inverse, and Hermitian. As an abuse of notation, we will frequently use $H_{\texttt{trans}}$ and H interchangeably.

A quantum algorithm A making q quantum queries to H is then specified by a sequence of unitary transformations $U_0, ..., U_q$. The evaluation of A then consists of alternately applying U_i and H to the initial state $|x\rangle$. We call

$$U_{i-1}H...U_1HU_0|x\rangle$$

the state of A before the ith query, and

$$HU_{i-1}H...U_1HU_0|x\rangle$$

the state after the ith query. The final state of the algorithm is

$$U_qH...U_1HU_0|x\rangle$$

We say that a quantum algorithm is efficient if q is a polynomial, and all the U_i are composed of a polynomial number of universal basis gates (the Hadamard, CNOT, and phase shift gates are commonly used).

Classical queries to an oracle are made by partial sampling over $\mathcal{X}$ before performing the query.

Density Matrix Suppose we have quantum states $|\phi_w\rangle$ for $w \in \mathcal{W}$, and we have a weight assignment D on $\mathcal{W}$. Then we define the density matrix

$$\rho = \sum_{w\in\mathcal{W}} D(x)|\phi_w\rangle\langle\phi_w|$$

The weight of an outcome y from measurement is given by $\langle y|\rho|y\rangle$. If D is a distribution, then this weight assignment is also a distribution, and the weight corresponds to the probability of obtaining y if we first pick w with probability $D(w)$, and then sample $|\phi_w\rangle$. In this case, the density matrix ρ summarizes the statistical behavior of the quantum system.

Suppose we have two weight assignments D_1 and D_2 on $\mathcal{W}$. Let D_1' and D_2' be the corresponding weight assignments of outcomes of measurement. Since the weight assignment induced by each $|\phi_w\rangle$ is in fact a distribution, $|D_1' - D_2'| \leq |D_1 - D_2|$.

For this paper, we will focus on the case where $\mathcal{W}$ is the set of oracles H from $X \to \mathcal{Y}$, and $|\phi_H\rangle$ is the final state of some quantum algorithm A that makes q queries to the oracle H.

B.1 Proof of Theorem 3.1

Proof of Theorem 3.1.

Let A be some quantum oracle algorithm. Suppose right before the first query, the state of the system is $|\phi_0\rangle = \sum_{xyz} \alpha_{xyz}|xyz\rangle$ (which is independent of the oracle). Assume for notational convenience that all the transition matrices U_i are identical and equal to U (the proof in the general case is essentially identical). Let $|\phi_{q,H}\rangle$ be the state of A after q queries to oracle H. That is,

$$|\phi_{q,H}\rangle = UHU....UH|\phi_0\rangle$$

Let ρ_q be the density matrix for A after q queries when the oracle H is drawn from a distribution D:

$$\rho_q = \sum_H \Pr_D[H]|\phi_{q,H}\rangle\langle\phi_{q,H}| = \sum_H \Pr_D[H]UHU...UH|\phi_0\rangle\langle\phi_0|HU^*...U^*HU^*$$

Observe that

$$\begin{aligned}(UH)_{xyzx'y'z'} &= \langle xyz|UH|x'y'z'\rangle \\ &= \langle xyz|U|x'y' \oplus H(x')z'\rangle = U_{xyz,x'y'\oplus H(x')z'}\end{aligned}$$

We can now evaluate ρ_q component-wise:

$$\begin{aligned}(\rho_q)_{xyz,x'y'z'} &= \sum_H \Pr_D[H](UHU...UH|\phi_0\rangle\langle\phi_0|HU^*...U^*HU^*)_{xyz,x'y'z'} \\ &= \sum_H \Pr_D[H]((UH)(UH)...(UH)|\phi_0\rangle\langle\phi_0|(UH)^*...(UH)^*(UH)^*)_{xyz,x'y'z'} \\ &= \sum_H \Pr_D[H] \sum_{x_qy_qz_q} \sum_{x_q'y_q'z_q'} \\ &\qquad U_{xyzx_qy_q\oplus H(x_q)z_q}((UH)...(UH)|\phi_0\rangle\langle\phi_0|(UH)^*...(UH)^*)U^*_{x'y'z'x_q'y_q'\oplus H(x_q')z_q'} \\ &\quad \vdots \\ &= \sum_H \Pr_D[H] \sum_{x_qy_qz_q} \sum_{x_q'y_q'z_q'} \cdots \sum_{x_1y_1z_1} \sum_{x_1'y_1'z_1'} \\ &\qquad U_{xyzx_qy_q\oplus H(x_q)z_q}U_{x_qy_qz_qx_{q-1}y_{q-1}\oplus H(x_{q-1})z_{q-1}}...U_{x_2y_2z_2,x_1y_1\oplus H(x_1)z_1}\alpha_{x_1y_1z_1} \\ &\qquad \alpha^*_{x_1'y_1'z_1'}U^*_{x_2'y_2'z_2',x_1'y_1'\oplus H(x_1')z_1'}...U^*_{x_q'y_q'z_q'x_{q-1}'y_{q-1}'\oplus H(x_{q-1}')z_{q-1}'}U^*_{x'y'z'x_q'y_q'\oplus H(x_q')z_q'}\end{aligned}$$

Now we can rearrange the order of summation as

$$\sum_{x_qy_qz_q} \sum_{x_q'y_q'z_q'} \cdots \sum_{x_1y_1z_1} \sum_{x_1'y_1'z_1'} \sum_H \Pr_D[H]$$

Next, notice that the summand only depends on $H(x_i), H(x'_i)$ for $i \in \{1,...,q\}$. This means, in the H sum, we can sum out all the inputs x for which $x \neq x_i, x'_i$. Letting $r_i = H(x_i), r'_i = H(x'_i)$, we have that the summation over H simplifies to

$$\sum_{r_1...r_q, r'_1...r'_q} \Pr_D[H(x_i) = r_i, H(x'_i) = r'_i \; \forall i \in \{1,...,q\}]$$

Putting it back together,

$$\begin{aligned}(\rho_q)_{xyz,x'y'z'} \quad = \quad & \sum_{x_q y_q z_q} \sum_{x'_q y'_q z'_q} \cdots \sum_{x_1 y_1 z_1} \sum_{x'_1 y'_1 z'_1} \sum_{r_1...r_q, r'_1...r'_q} \\ & \Pr_D[H(x_i) = r_i, H(x'_i) = r'_i \; \forall i \in \{1,...,q\}] \\ & U_{xyzx_q y_q \oplus r_q z_q} U_{x_q y_q z_q x_{q-1} y_{q-1} \oplus r_{q-1} z_{q-1}} ... U_{x_2 y_2 z_2, x_1 y_1 \oplus r_1 z_1} \alpha_{x_1 y_1 z_1} \\ & \alpha^*_{x'_1 y'_1 z'_1} U^*_{x'_2 y'_2 z'_2, x'_1 y'_1 \oplus r'_1 z'_1} ... U^*_{x'_q y'_q z'_q x'_{q-1} y'_{q-1} \oplus r'_{q-1} z'_{q-1}} U^*_{x' y' z' x'_q y'_q \oplus r'_q z'_q}\end{aligned}$$

Thus, the density matrix ρ_q, which contains all the statistical information about the algorithm A, is a linear combination of the quantities $\Pr_D[H(x_i) = r_i, H(x'_i) = r'_i \; \forall i \in \{1,...,q\}]$. In particular, the probability of any particular output is on the diagonal of ρ_q, and is thus a linear combination. Since the probabilities are real, we can take the coefficients of $\Pr_D[H(x_i) = r_i, H(x'_i) = r'_i \; \forall i \in \{1,...,q\}]$ to be real as well. □

C More on Semi-Constant Distributions

In this section, we prove Lemma 4.2, which states that the marginal distributions for any k inputs of oracles drawn from SC_λ specified by polynomials of degree k for which the λ^1 coefficient is 0. We also describe a quantum algorithm for finding collisions in SC_λ using $O(\lambda^{-2/3})$ oracle queries.

C.1 Proof of Lemma 4.2

Proof of Lemma 4.2. Recall that SC_λ is defined as follows:

- Pick y at random from $\mathcal{Y}$.
- For each $x \in \mathcal{X}$, with probability $1 - \lambda$, set $H(x)$ to be a random element of $\mathcal{Y}$. With probability λ, set $H(x) = y$.

Suppose $\mathcal{Y}$ contains N elements. Let $\alpha((x_j),(r_j)) = \Pr[H(x_i) = r_j \forall i \in \{1,...,k\}]$, the probability that x_j maps to r_j for k values of x_j and r_j. Suppose there are ℓ distinct r_j, denoted t_m, and let k_m be the number of r_j equal to t_m (note that $\sum_{m=1}^{\ell} k_m = k$). Let $\mathcal{F}$ be the set $\{1,...,\ell,\perp\}$. For each $f \in \mathcal{F}$, if $f = \perp$, f is associated with the event that $t_m \neq y$ for all m, and otherwise, f is associated with the event that $t_f = y$.

Let $\mathtt{eq}_{\mathtt{f},\mathtt{m}}$ be 1 if $f = m$, 0 otherwise. The probability that $O(x_j) = t_m$ is $1/N$ if we are choosing the output at random, and otherwise, it is 1 if $t_m = y$ and 0 otherwise. In other words, this probability is

$$(1-\lambda)\frac{1}{N} + \lambda \mathtt{eq}_{f,m} = \frac{1}{N} + \lambda\left(\mathtt{eq}_{f,m} - \frac{1}{N}\right) .$$

Since there are k_m copies of each t_m among the r_j,

$$\Pr[H(x_i) = r_j \forall i \in \{1, ..., k\} | f] = \prod_{i=m}^{\ell} \left(\frac{1}{N} + \lambda \left(\mathsf{eq}_{f,m} - \frac{1}{N} \right) \right)^{k_m} .$$

Summing over all f gives

$$\alpha((x_i), (r_j)) = \sum_f \Pr[f] \prod_{i=m}^{\ell} \left(\frac{1}{N} + \lambda \left(\mathsf{eq}_{f,m} - \frac{1}{N} \right) \right)^{k_m} .$$

This is a polynomial in λ. It is a sum of products of $\sum k_m = k$ monomials in λ, so its total degree is at most k. Now, we shall approximate this to first order in λ, and show that the first-order coefficient is 0:

$$\begin{aligned}
\alpha((x_i), (r_j)) &= \sum_f \Pr[f] \prod_{m=1}^{\ell} \left(\frac{1}{N} + \lambda \left(\mathsf{eq}_{f,m} - \frac{1}{N} \right) \right)^{k_m} \\
&= \sum_f \Pr[f] \prod_{m=1}^{\ell} \left(\frac{1}{N^{k_m}} + k_m \frac{1}{N^{k_m - 1}} \lambda \left(\mathsf{eq}_{f,m} - \frac{1}{N} \right) \right) + O(\lambda^2) \\
&= \frac{1}{N^{\sum_m k_m}} \sum_f \Pr[f] \prod_{m=1}^{\ell} \left(1 + k_m N \lambda \left(\mathsf{eq}_{f,m} - \frac{1}{N} \right) \right) + O(\lambda^2) \\
&= \frac{1}{N^k} \sum_f \Pr[f] \left(1 + N\lambda \sum_{m=1}^{\ell} k_m \left(\mathsf{eq}_{f,m} - \frac{1}{N} \right) \right) + O(\lambda^2) \\
&= \frac{1}{N^k} + \frac{\lambda}{N^{k-1}} \sum_{m=1}^{\ell} k_m \left(\sum_f \Pr[f] \mathsf{eq}_{f,m} - \frac{1}{N} \right) + O(\lambda^2)
\end{aligned}$$

Notice that $\sum_f \Pr[f] \mathsf{eq}_{f,m}$ is the probability that $t_m = y$, which is equal to $1/N$. Thus, all the terms in the first-order coefficient cancel out, so the λ^1 coefficient is 0. This completes the lemma. □

C.2 Finding Collisions in Semi-Constant Distributions

Here we explore the problem of finding a collision in an oracle drawn from a semi-constant distribution SC_λ over $\mathcal{H}_{\mathcal{X},\mathcal{Y}}$. Our motivation for studying the collision problem is as follows: a classical algorithm making q_H queries can only distinguish SC_λ from random with probability $O(\lambda^2 q_H^2)$, since it can only distinguish the two cases if it happens to query two distinguished points. Further, querying random points, and outputting 1 if a collision is found achieves this bound. Thus, a collision search is the best way to distinguish SC_λ from random in the classical setting, and the same may also be true in the quantum setting

Let $N = |\mathcal{Y}|$ be the number of elements in $\mathcal{Y}$, and assume that $\lambda >> \frac{1}{N}$ so that finding a collision requires (with high probability) finding a collision in the distinguished inputs.

Let c be the minimum constant such that SC_λ is indistinguishable from uniform with except with probability $O(q^c \lambda^2)$. Corollary 4.3 shows us that $c \leq 4$. We will now show that $c \geq 3$, using

the following algorithm. The algorithm is basically the algorithm of Brassard et al. [BHT97], but modified for our purposes. It operates as follows:

- Let $p = (q-1)/2)$. Select a subset $\mathcal{W} \subseteq \mathcal{X}$ of size p. For each $x \in \mathcal{W}$, store the pair $(x, H(x))$ in a table, sorted by the second coordinate. Check if there is a collision in $\mathcal{W}$. If so, output this collision.
- Construct the oracle $O(x)$ which is 1 if and only if $x \notin \mathcal{W}$ and $H(x) = H(x_0)$ for some $x_0 \in \mathcal{W}$. Since the entries to $\mathcal{W}$ are sorted by the second coordinate, this test can be performed efficiently.
- Run Grover's algorithm on this oracle for p queries, looking for an $x \notin \mathcal{W}$ such that $H(x) = H(x_0)$ for some $x_0 \in \mathcal{W}$.
- Output (x, x_0).

The first step uses p queries. If the algorithm is given an oracle from SC_λ, the probability that we find a distinguished input in the first step is

$$1 - (1-\lambda)^{|\mathcal{W}|} = 1 - (1-\lambda)^p \geq p\lambda$$

Grover's algorithm [Gro96] finds an x such that $O(x) = 1$ with probability $\Omega(p^2 f)$ using p queries, where f is the fraction of inputs to O that map to 1. Thus, f is the fraction of x such that $x \notin \mathcal{W}$ (which is most of them) and $H(x) = H(x_0)$ for some $x_0 \in \mathcal{W}$. If we found a distinguished input, this fraction is (in expectation) at least λ.

If we found a distinguished input in the first step (which happens with probability at least $p\lambda$), Grover's algorithm will find an x such that $O(x) = 1$ with probability $\Omega(p^2\lambda)$ using p queries. Thus, out algorithm uses $2p$ queries, and finds a collision with probability $\Omega(p^3\lambda^2)$.

If this algorithm is given a random oracle instead, the probability that we find a collision is negligible. By adding an extra check that the output of the algorithm is indeed a collision (requiring 1 extra query since $H(x_0)$ has already been recorded), we get an algorithm using $2p+1 = q$ quantum queries that distinguishes SC_λ from random with probability $\Omega(p^3\lambda^2) = \Omega(q^3\lambda^2)$, showing that $c \geq 3$.

D Proof of Claim 1

Here we complete the proof of Theorem 5.1 by proving Claim 1.

Proof of Claim 1. Recall the setup: A' makes $q \equiv q_H + q_S + 1$ queries to an oracle H drawn from SC_λ, and outputs (c, id^*, Q). Let $\mathcal{X}$ be the set of distinguished inputs to H. If $\mathsf{id}^* \in \mathcal{X}$ and $Q \bigcap \mathcal{X} = \emptyset$, A' wins if $c = 0$. Otherwise, we flip a coin to decide if A' wins. Let $p(\lambda)$ be the probability that A' wins. We want to show that $p(\lambda) \geq \frac{1}{2} + \lambda\epsilon - \frac{\ell(q_H,q_S)}{4}\lambda^2$ for some polynomial $\ell(\cdot,\cdot)$, where ϵ is the probability that $c = 0$ when A' is given a truly random oracle, minus $frac12$. We assume without loss of generality that A' never outputs $\mathsf{id}^* \in Q$.

By Theorem 3.1,

$$\Pr_{H \leftarrow D}[A'^H() = (c, \mathsf{id}^*, Q)] = \sum_{x_1,\ldots,x_{2q},r_1,\ldots,r_{2q}} \beta(c, \mathsf{id}^*, Q, (x_i), (r_i)) \Pr_{H \leftarrow D}[H(x_i) = r_i \forall i \in \{1,\ldots,2q\}]$$

for some coefficients $\beta(c, \mathsf{id}^*, Q, (x_i), (r_i))$. Then notice that

$$\frac{1}{2} + \epsilon = \sum_{\mathsf{id}^*, Q} \sum_{x_1,...,x_{2q}, r_1,...,r_{2q}} \beta(0, \mathsf{id}^*, Q, (x_i), (r_i)) \Pr_H[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$$

where H is a random oracle.

Call (id^*, Q) good if $\mathsf{id}^* \in \mathcal{X}$ and $Q \bigcap \mathcal{X} = \emptyset$, and bad otherwise. For each c, let γ_c be the probability of obtaining (c, id^*, Q) such that (id^*, Q) is good. The probability A' wins is then:

$$p(\lambda) = \gamma_0 + (1 - \gamma_0 - \gamma_1)\frac{1}{2} = \frac{1}{2}(1 + \gamma_0 - \gamma_1) \ .$$

Fix a set $\mathcal{X}$, and let $\mathsf{SC}_{\mathcal{X}}$ be the distribution on oracles H where a y is chosen at random, and for each $x \in \mathcal{X}$, $H(x) = y$, and for each $x \notin \mathcal{X}$, $H(x)$ is chosen at random.

We can now compute γ_c:

$$\gamma_c = \sum_{\mathcal{X}} \Pr[\mathcal{X}] \sum_{\mathsf{id}^* \in \mathcal{X}} \sum_{Q : Q \bigcap \mathcal{X} = \emptyset} \Pr_{H \leftarrow \mathsf{SC}_{\mathcal{X}}}[A^H() = (c, \mathsf{id}^*, Q)]$$

where $\Pr[\mathcal{X}]$ is the probability that $\mathcal{X}$ is chosen as the set of distinguished inputs, and Q ranges over sets of size q_S. We can rewrite γ_c as

$$\begin{aligned}
\gamma_c &= \sum_{\mathsf{id}^*} \sum_{\substack{Q: \\ \mathsf{id}^* \notin Q}} \left(\sum_{\substack{\mathcal{X}: \\ \mathsf{id}^* \in \mathcal{X}, Q \bigcap \mathcal{X} = \emptyset}} \Pr[\mathcal{X}] \Pr_{H \leftarrow \mathsf{SC}_{\mathcal{X}}}[A^H() = (c, \mathsf{id}^*, Q)] \right) \\
&= \sum_{\mathsf{id}^*} \sum_{\substack{Q: \\ \mathsf{id}^* \notin Q}} \left(\sum_{\substack{\mathcal{X}: \\ \mathsf{id}^* \in \mathcal{X}, Q \bigcap \mathcal{X} = \emptyset}} \Pr[\mathcal{X}] \sum_{\substack{x_1,...,x_{2q} \\ r_1,...,r_{2q}}} \beta(c, \mathsf{id}^*, Q, (x_i), (r_i)) \Pr_{H \leftarrow \mathsf{SC}_{\mathcal{X}}}[H(x_i) = r_i \forall i] \right) \\
&= \sum_{\mathsf{id}^*} \sum_{\substack{Q: \\ \mathsf{id}^* \notin Q}} \left(\sum_{\substack{x_1,...,x_{2q} \\ r_1,...,r_{2q}}} \beta(c, \mathsf{id}^*, Q, (x_i), (r_i)) \sum_{\substack{\mathcal{X}: \\ \mathsf{id}^* \in \mathcal{X}, Q \bigcap \mathcal{X} = \emptyset}} \Pr[\mathcal{X}] \Pr_{H \leftarrow \mathsf{SC}_{\mathcal{X}}}[H(x_i) = r_i \forall i] \right)
\end{aligned}$$

We now look at the quantity

$$\sum_{\substack{\mathcal{X}: \\ \mathsf{id}^* \in \mathcal{X}, Q \bigcap \mathcal{X} = \emptyset}} \Pr[\mathcal{X}] \Pr_{H \leftarrow \mathsf{SC}_{\mathcal{X}}}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}] \ . \tag{D.1}$$

Since $\Pr_{\mathcal{X} \leftarrow \mathsf{SC}_{\mathcal{X}}}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}]$ only depends on the inputs x_i, and each x is put in $\mathcal{X}$ with independent probability λ, we can sum out all x other than id^*, x_i, or those in Q. Therefore, we only consider $\mathcal{X}$ drawn as a subset of $\{x_i : i \in \{1, ..., 2q\}\} \bigcup Q \bigcup \{\mathsf{id}^*\}$.

Let $\mathcal{X}'$ be drawn from the subsets of $\{x_i : i \in \{1, ..., 2q\}, x_i \notin Q, x_i \neq \mathsf{id}^*\}$ where each element is in $\mathcal{X}'$ with independent probability λ. We can then rewrite $\Pr[\mathcal{X}]$ as the probability that $\mathsf{id}^* \in \mathcal{X}$ (which is λ) times the probability that Q is disjoint from $\mathcal{X}$ (which is $(1-\lambda)^{q_S}$) times the probability of drawing $\mathcal{X}'$, where $\mathcal{X}' = \mathcal{X} \setminus \{\mathsf{id}^*\}$. Thus, (D.1) becomes

$$\lambda(1 - \lambda)^{q_S} \sum_{\mathcal{X}'} \Pr[\mathcal{X}'] \Pr_{H \leftarrow \mathsf{SC}_{\mathcal{X}' \cup \{\mathsf{id}^*\}}}[H(x_i) = r_i \forall i \in \{1, ..., 2q\}] \tag{D.2}$$

Now, each $\Pr[X']$ has the form $\lambda^c(1-\lambda)^d$, where c is the number of elements in $\mathcal{X}'$, and d is the number of elements left out. Thus $c+d \leq 2q$, so $\Pr[X']$ is a polynomial of degree at most $2q$ in λ. Thus, (D.2) is a polynomial of degree at most $2q+q_S+1$. Moreover, if we let $\lambda \to 0$, $\Pr[X'] \to 0$, and thus (D.2) becomes

$$\lambda \Pr_{H \leftarrow \mathsf{SC}_{\{\mathsf{id}^*\}}}[H(x_i) = r_i \forall i \in \{1,...,2q\}]$$

Notice that $\mathsf{SC}_{\{\mathsf{id}^*\}}$ is drawn as follows: pick a random y, and set $H(\mathsf{id}^*) = y$, for all other x, choose $H(x)$ randomly. This is identical to a truly random oracle, so (D.2) is equal to

$$\lambda \Pr_H[H(x_i) = r_i \forall i \in \{1,...,2q\}] + O(\lambda^2) \ .$$

Plugging this into our expression for γ_c, we get that γ_c is a polynomial of degree at most $2q+q_S+1$, and that it is equal to

$$\gamma_c = \sum_{\mathsf{id}^*} \sum_{\substack{Q: \\ \mathsf{id}^* \notin Q}} \left(\sum_{\substack{x_1,...,x_{2q} \\ r_1,...,r_{2q}}} \beta(c, \mathsf{id}^*, Q, (x_i), (r_i)) \Pr_H[H(x_i) = r_i \forall i \in \{1,...,q\}] \right) \lambda + O(\lambda^2)$$

Notice that the first-order coefficient is the probability that the first bit of the output of A' is c when given a truly random oracle. Thus:

$$\begin{aligned} \gamma_0 &= \left(\frac{1}{2} + \epsilon\right)\lambda + O(\lambda^2) \\ \gamma_1 &= \left(\frac{1}{2} - \epsilon\right)\lambda + O(\lambda^2) \end{aligned}$$

Thus, the probability A' wins is

$$p(\lambda) = \frac{1}{2}(1 + \gamma_0 - \gamma_1) = \frac{1}{2} + \epsilon\lambda + O(\lambda^2)$$

and is specified by a polynomial of degree at most $2q+q_S+1$. Since this probability is always between 0 and 1 for all $\lambda \in [0,1]$, we can use the Markov brother's inequality [DS41] to bound the magnitude of the second derivative to at most $|p''(\lambda)| \leq \frac{2}{3}(2q+q_S+1)^4$. Integrating, we can bound the first derivative to

$$p'(\lambda) \geq p'(0) - \frac{2}{3}(2q+q_S+1)^4\lambda = \epsilon - \frac{2}{3}(2q+q_S+1)^4\lambda \ .$$

Performing the integration once more, we get that

$$p(\lambda) \geq p(0) + \epsilon\lambda - \frac{1}{3}(2q+q_S+1)\lambda^2 = \frac{1}{2} + \epsilon\lambda - \frac{\ell(q_H,q_H)}{4}\lambda^2$$

where $\ell(q_H,q_S) = \frac{4}{3}(2q_H+3q_S+2)^4$.

□

E Proof of Theorem 5.3

We prove Theorem 5.3, which states that the FDH signature scheme is secure in the quantum random oracle model if the underlying trapdoor permutation is secure against quantum adversaries.

Proof of Theorem 5.3. Suppose towards contradiction that there is a quantum adversary A making q_H hash queries, q_S signature queries, that breaks S with probability ϵ. Assume without loss of generality that A never tries to forge a signature on a message it already received a secret key for.

Let Game_0 be the standard attack game for S: the challenger generates $(\mathsf{sk}, \mathsf{pk})$ from $\mathsf{S.Gen}$, and sends pk to the adversary. The adversary can make (classical) signature queries on messages m_i, to which the challenger responds with $\mathsf{S.Sign}^H_{\mathsf{sk}}(m_i)$, and (quantum) hash queries to the random oracle H. A wins if it can produce a pair (m^*, σ^*) such that $m^* \neq m_i$ for any i, and $\mathsf{S.Ver}^H_{\mathsf{pk}}(m^*, \sigma^*) = \mathsf{accept}$. The success probability in Game_0 is ϵ.

Assume without loss of generality that A never asks for a signature on m^*. Let A' be the following algorithm that makes quantum queries to an oracle H, and simulates the interaction between A and the challenger: generate $(\mathsf{sk}, \mathsf{pk})$ from $\mathsf{S.Gen}$, send pk to A. Run A, forwarding all hash queries to H, and answering extraction queries using $\mathsf{S.Sign}^H$. The output of A' is $(\mathsf{S.Ver}^H_{\mathsf{pk}}(m^*, \sigma^*), m^*, \sigma^*, Q)$ where Q is the list of signature queries made by A. We can now think of Game_0 as follows: run A' with a random oracle H to obtain (m^*, σ^*, Q). Report that the game is won if and only if $\mathsf{S.Ver}^H_{\mathsf{pk}}(m^*, \sigma^*) = \mathsf{accept}$. The number of queries to H made by A' is q_H for direct queries, q_S for queries through the signature algorithm, and 1 for the check $\mathsf{S.Ver}^H_{\mathsf{pk}}(m^*, \sigma^*)$, for a total of $q_H + q_S + 1$ queries.

Let $\lambda \in (0, 1)$ to be chosen later. Let $\mathcal{M}$ be the message space. Let Game_1 be Game_0, except that we choose a subset $\mathcal{X}$ of $\mathcal{M}$ as follows: for each $m \in \mathcal{M}$, we put m in $\mathcal{X}$ with probability λ. If A asks for a signature on a message in $\mathcal{X}$, abort and A loses. If the message A attempts to forge is *not* in $\mathcal{X}$, also abort and A loses. Otherwise, follow the criteria in Game_0 to decide if A wins. The message A tries to forge must be distinct from the messages it received signatures for, so the probabilities that each of these messages are in $\mathcal{X}$ is independent. Therefore, the no-abort conditions are met with probability at least $\lambda(1 - \lambda q_S)$. Thus, A wins in Game_1 with probability at least $\lambda(1 - \lambda q_S)\epsilon \geq \lambda\epsilon - q_S\lambda^2$.

Let Game_2 be Game_1, except that we choose a random y in the public key space of S, and set $H(x) = y$ for all x in $\mathcal{X}$. H is now distributed according to SC_λ, and by the same arguments as in the proof of Theorem 5.1, the probability A wins in Game_2 is at least $\lambda\epsilon - \frac{\ell(q_H, q_S)}{4}\lambda^2$ for some polynomial $\ell(\cdot, \cdot)$.

Let us now restate Game_2 in terms of A: pick $\mathcal{X}$ and H as above, and perform the standard attack game for S using H as the random oracle. When A produces (m^*, σ^*), first check that no signature queries were in $\mathcal{X}$ and that the message m^* produced by A *is* in $\mathcal{X}$. If so, A wins if and only if $\mathsf{S.Ver}^H_{\mathsf{pk}}(m^*, \sigma^*) = \mathsf{accept}$. Otherwise, if the check fails, A loses. We can now make the following observation: once an extraction query in $\mathcal{X}$ is made, we can abort running A, since we no longer need A to determine if the game is won.

Now we are ready to define an algorithm B what inverts f. Give B access to two random oracles O_1 and O_2. O_1 maps messages to inputs to f. O_2 maps messages to bits, where the probability of outputting 1 is λ. On input (pk, y), B works as follows:

- Send pk to A, and simulate A, playing the role of challenger to A.

- Construct the (quantum) oracle H such that

$$H(m) = \begin{cases} y & \text{if } O_2(m) = 1 \\ f_{\mathsf{pk}}(O_1(m)) & \text{otherwise} \end{cases}$$

- When A makes a signature query on a message m_i, compute $O_2(m_i)$. If the result is 1, abort. Otherwise, return $\sigma_i = O_1(m_i)$.
- When A returns a forgery candidate (m, σ), test if $O_2(m) = 1$ and $f_{\mathsf{pk}}(\sigma) = y$. If so, output σ. Otherwise, abort.

Using an analysis similar to that of Theorem 5.1, we get that the advantage of B is at least

$$\frac{\epsilon^2}{\ell(q_H, q_S)}$$

when we set $\lambda = 2\epsilon/\ell(q_H, q_S)$. □