Tracing Quantum State Distinguishers via Backtracking

Mark Zhandry
NTT Research
mzhandry@gmail.com

Abstract

We show the following results:

- The post-quantum equivalence of indistinguishability obfuscation and differing inputs obfuscation in the restricted setting where the outputs differ on at most a polynomial number of points. Our result handles the case where the auxiliary input may contain a *quantum state*; previous results could only handle classical auxiliary input.
- Bounded collusion traitor tracing from general public key encryption, where the decoder is allowed to contain a *quantum state*. The parameters of the scheme grow polynomially in the collusion bound.
- Collusion-resistant traitor tracing with constant-size ciphertexts from general public key encryption, again for *quantum state decoders*. The public key and secret keys grow polynomially in the number of users.
- Traitor tracing with embedded identities in the keys, again for *quantum state decoders*, under a variety of different assumptions with different parameter size trade-offs.

Traitor tracing and differing inputs obfuscation with quantum decoders / auxiliary input arises naturally when considering the post-quantum security of these primitives. We obtain our results by abstracting out a core algorithmic model, which we call the Back One Step (BOS) model. We prove a general theorem, reducing many quantum results including ours to designing *classical* algorithms in the BOS model. We then provide simple algorithms for the particular instances studied in this work.

1 Introduction

The threat of quantum computers requires re-evaluation of some of the core aspects of modern cryptography. Due to Shor's algorithm [Sho94], cryptosystems based on factoring or discrete logs will be insecure. Even for cryptosystems based on so-called "post-quantum" building blocks, quantum computing may also yield new threat models, such as superposition attacks [KM10, Zha12, DFNS14].

A perhaps more subtle issue is the following: even if the building blocks are quantum-immune and the threat model remains the same, the classical proof may not hold quantumly. Indeed, a classical proof converts a *classical* adversary into a *classical* algorithm for some underlying hard problem, while a post-quantum proof must convert a *quantum* adversary into a *quantum* algorithm. What works for classical adversaries may not work for quantum adversaries. The canonical example of a proof technique that does not translate is rewinding, which is known to be problematic quantumly

due to the no-cloning theorem [VDG98, ARU14][1]. Rewinding is most often discussed in the context of interactive proofs, where it is used to extract information from the adversary that would remain hidden against straight-line procedures. In this context, a number of positive results have been achieved quantumly [Wat06, Unr12, CMSZ21, LMS21].

In this work, we consider certain cases where rewinding arises, perhaps implicitly, in settings other than interactive proofs:

- It is known, classically, that indistinguishability obfuscation (iO) implies differing inputs obfuscation (diO) in the setting where the pairs of circuits being considered only differ on polynomially many points [BCP14]. The proof extracts a differing input from a distinguisher by testing the distinguisher on a variety of distributions over programs, using a type of binary search.

- In traitor tracing [CFN94], a coalition of malicious users group their secret keys into a pirate decoder program, and a tracing algorithm must identify at least one malicious user. Once identified, the malicious user(s) can be prosecuted. Essentially all traitor tracing schemes test the decoder on various distributions over ciphertexts, and use the corresponding decryption probabilities to accuse a user.

The unifying feature of both classes of results is that the extracted information is computed based on the success probabilities of a single adversary on various distributions of inputs. This is very different from the way information is typically extracted for interactive proofs, where the information is usually extracted from the adversary's outputs themselves.

When moving to the quantum setting for traitor tracing, Zhandry [Zha20] shows that if the pirate decoder contains a quantum state, then the classical tracing algorithms are no longer guaranteed to work. The malicious users of course want to design their decoder in such a way as to avoid tracing, and would therefore be incentivized to design a quantum state decoder to evade tracing. Being able to trace such quantum state decoders is therefore the natural model to consider in the post-quantum setting. Zhandry provides an initial positive result, showing how to trace even quantum decoders for schemes in the Private Linear Broadcast Encryption (PLBE) framework [BSW06], when the identity space is polynomial size[2]. However, the techniques are incapable of handling other important traitor tracing approaches, in particular:

- Traitor tracing with embedded identities, where the identity of a user is an arbitrary string, as opposed to an index in a polynomial-sized set. The first such tracing scheme is due to [NWZ16], who use the PLBE framework but with *exponentially many* identities. [NWZ16] and later [GKW19] also explore other structures for achieving embedded identities.

- Combinatorial traitor tracing, such as [CFN94, BN08, BP08], which achieve traitor tracing with short ciphertexts from general public key encryption, as opposed to algebraic tools.

When considering the restricted iO-diO equivalence in the quantum setting, it is most natural to consider a quantum auxiliary input, since the auxiliary input will be the adversary's state at some step in the protocol. However, the equivalence has a flavor similar to PLBE with super-polynomial

[1] A different example is the quantum random oracle model [BDF+11], though this model is conceptually closer to the superposition attacks mentioned above.

[2] Another limitation of Zhandry's work is that the PLBE must support public encryption for all distributions used during tracing, which is not true of the known succinct LWE-based scheme [GKW18].

identity spaces (as observed by [NWZ16]), and therefore is also not handled by the previous quantum techniques.

Thus, these important cases of traitor tracing and the iO-diO equivalence were previously open questions in the quantum decoder/auxiliary input setting.

This Work. In our work, we resolve these open questions. We start by defining an algorithmic model we call the Back One Step (BOS) model, which we show can be realized by any quantum program containing a quantum state. This is the core conceptual contribution of this work, and builds on ideas from Zhandry [Zha20] and Chiesa et al. [CMSZ21], which in turn build on Marriott and Watrous [MW04].

We then design new algorithms for the BOS model, for the instances arising from the aforementioned open problems. This step is entirely classical, and while the algorithms may not be trivial, they are fairly simple. Combining our results together, we achieve the following results:

- (Section 6) We prove the restricted iO-diO equivalence holds post-quantumly, even if the auxiliary input is a quantum state. Along the way, we give a definition of diO that address several subtleties in defining quantum security that we overcome.
- We construct tracing algorithms for several existing traitor tracing schemes, achieving a couple of "firsts" for tracing schemes for quantum decoders:
  - (Section 8) Bounded collision traitor tracing from general public key encryption, where the parameters grow polynomially with the collusion bound but are independent of the number of users.
  - (Section 8) Collusion-resistant traitor tracing from general public key encryption, where ciphertext size is independent of the number of users.
  - (Sections 7 and 9) Collusion-resistant *embedded identity* traitor tracing, under various assumptions with different parameter size trade-offs. In particular, assuming public key encryption, we get a scheme where the parameters grow polynomially in the number of users. Assuming Learning With Errors (LWE), we get such a scheme with succinct ciphertexts whose length is independent of the length of the embedded identities. Finally, assuming iO, we get a scheme with both succinct ciphertexts and where parameter sizes are independent of the number of users.

  These schemes are identical to their classical counterparts, *except* for the tracing algorithm. They also match most of the best-known results for classical traitor tracing under post-quantum assumptions, with the main exception being the LWE-based traitor tracing scheme with constant-sized parameters [GKW18], which still remains open in the quantum setting.
- (Section 10) We show limitations of the BOS model, showing an artificial setting that can be solved classically, but for which there is no BOS model algorithm.

Acknowledgements

We thank Fermi Ma for helpful discussions.

2 Technical Overview

2.1 A general view of classical tracing algorithms

Essentially all modern classical traitor tracing algorithms can be framed as follows. There is a collection $\mathcal{D} = \{D_q\}_{q\in\mathcal{Q}}$ of distributions of ciphertexts for some index set $\mathcal{Q}$, and the adversary produces a decoder subject to the following guarantees:

- **Large success probability on source.** The distribution of "honest" ciphertexts is an element of $\mathcal{D}$, which we will denote as D_α for some distinguished $\alpha \in \mathcal{Q}$ that we call the source. Any useful decoder, by definition, has a "large" success probability on D_α.
- **Small success probability on sinks.** There is a collection of distributions $\{D_q\}_{q\in\Omega}$ for $q \in \Omega \subset \mathcal{Q}$, which we may call sinks, for which any coalition of users has "small" success probability.
- **Constant on adversary partition.** Any coalition of malicious users corresponds to a partition Π on $\mathcal{Q}$, such that the coalition, and therefore their decoder, cannot efficiently distinguish between distributions belonging to the same part of the partition. In other words, the decoder's decryption probability is roughly constant on each part of the partition.

The tracing algorithm therefore tests the decoder on various D_q, learning (estimates of) the decoder's success probabilities on D_q. The rough idea is that α and $q \in \Omega$ must have different success probabilities, so the tracer is guaranteed to see some jumps as it makes its queries. These jumps must be across different parts of the partition, revealing some information about the structure of the partition. The hope is that this information can identify at least one malicious user.

Abstracting out the details of generating the various distributions and estimating their success probability, we arrive at a model we might call the Globally Consistent Hidden Partition model, where the tracer simply queries (potentially adaptively) on various $q_i \in \mathcal{Q}$ and receives in response real numbers $o_i \in [0, 1]$. We are guaranteed that any o_i, o_j which correspond to queries q_i, q_j lying in the same part of the partition Π will be close, that $q_i = \alpha$ will give "large" o_i, and that $q_i \in \Omega$ will give "small" o_i. The tracer then takes these o_i and uses them to learn information about the partition.

Example : PLBE. To make things concrete, we now illustrate how this view captures private linear broadcast encryption (PLBE). Here, there are N users, with identities $1, \ldots, N$. For any $q \in [0, N]$, one can encrypt to the set $[q] = \{1, \ldots, q\}$. Users in $[q]$ will be able to decrypt, while users outside of $[q]$ will not. An "honest" ciphertext is an encryption to all users, $q = N$. Moreover, any coalition of users will be unable to distinguish encryptions to $[a]$ and $[b]$, unless the coalition contains a user in the half-open interval $(a, b]$. In the view above, we would therefore have $\mathcal{Q} = [0, N]$, $\alpha = N$, and $\Omega = \{0\}$. For a coalition of users $a_1 < a_2 < \cdots < a_t \in [N]$, the partition Π therefore consists of the intervals $[0, a_1), [a_1, a_2), [a_2, a_3), \cdots, [a_t, N]$.

To trace, we split into two cases. In the case where N is polynomial, one simply estimates the success probabilities $o_0, o_1, \ldots, o_N$ on queries $0, 1, \cdots, N$. The usefulness of the decoder implies that $o_N \gg 0$, while PLBE security implies that $o_0 \approx 0$. Therefore, there must exist a q such that $|o_{q-1} - o_q| \gg 0$. But since o_i must be constant on the intervals in Π, we know that $q = a_i$ for some i. Thus we have identified a member of the coalition.

If N is exponential, then we cannot simply do a linear scan. However, a variant of a binary search will work, as shown in [BCP14, NWZ16]. In the binary search, we recurse left if there is a large gap between the pivot and the left value, while we recurse right if there is a large gap between the pivot and the right value. Importantly, we must make sure to pivot *both* left and right if there is a large gap in both directions. Otherwise, there is a possibility that every time we recurse, the gap gets divided by 2, and after a polynomial number of steps the gap is negligible but we have yet to accuse a user. Fortunately, it is proved by those works that there will never be more branches in the binary search than there are users in the coalition, so the overall search still takes polynomial time.

2.2 Moving to Quantum: Prior Results

When we move to allowing a quantum decoder, we must be much more careful. There are several issues, first pointed out by Zhandry [Zha20]. First is definitional: the success probability of the decoder is not necessarily known (nor even *knowable*) until it is *measured*, and any measurement necessarily alters the quantum state. Care must be taken to appropriately define traitor tracing to handle cases where the decoder may be in superposition of both high success probability and low success probability. Zhandry shows how to handle these definitional issues.

The second issue has to do with rewinding, which happens at two levels. First, estimating success probability on a distribution classically involves running the decoder on various samples from the distribution. But this implicitly requires being able to return the decoder to its original state every time, to ensure independent trials from the same distribution. As mentioned above, quantumly rewinding is problematic. In particular, each trial may alter the state of the decoder. Zhandry demonstrates that the classical approach of repeated sampling cannot, in general, yield an estimate of the decoder's success probability. Fortunately, Zhandry shows how to develop a quantum-compatible success probability estimation routine based on a technique of Marriott and Watrous [MW04].

Remark 2.1. Note that Zhandry's procedure requires security to hold when the adversary gets an arbitrary polynomial number of tracing ciphertexts; for traitor tracing from LWE [GKW18], this is not guaranteed. We inherit this limitation, and this is why we still cannot handle succinct traitor tracing from LWE.

A second level is that multiple success probabilities will have to be estimated, for various distributions of inputs. In the classical setting, it is assumed that each estimate of success probability is applied to the same initial decoder; in other words, the decoder is rewound to its initial state[3]. Otherwise, how can we make sense of gaps between estimated success probabilities, if the success probabilities has since changed and the previous estimates are no longer valid?

Zhandry gives a first step toward overcoming this problem, by showing that a *local* consistency property holds: if two *consecutive* estimates of success probabilities are on computationally indistinguishable distributions, then the estimated success probabilities will be close. However, estimated probabilities that are not consecutive may not obey this property. This gives rise to a variant of the tracing model above, which we will call the *Locally Consistent Hidden Partition Model*, that is identical to the Globally Consistent model above, except that (1) o_i, o_j corresponding to q_i, q_j in the same part are guaranteed close *only if* $|i - j| \leq 1$, and (2) $q_i = \alpha$ is guaranteed to yield a large r_i *only on the first query* $i = 1$. The good news is that the linear scan algorithm for PLBE

[3]The usual terminology is that the decoder is "stateless."

with polynomial identity space fits within this model (provided the scan starts at N and goes to 0), allowing Zhandry to trace this particular case. On the other hand, the model seems inherently limited to PLBE: during any sequence of probability estimates, the estimate outcomes may be the sequence $1, 1, 1, \cdots, 1, 0, 0, 0, \cdots$. Such a sequence would satisfy the local consistency requirements, as long as the jump from 1 to 0 happens the first time a distribution D_q for $q \in \Omega$ is tested. But for a polynomial-length sequence, only a logarithmic number of bits of information can be extracted, namely the location of the jump between 1 and 0. Thus this model is incapable of handling a variety of traitor tracing scenarios, such as

- Embedded identities, where the whole point is to embed more than a logarithmic amount of information.
- Combinatorial traitor tracing, which usually follows a non-linear path. For example, traitor tracing based on fingerprinting codes [BN08, BP08] first extracts a very long "codeword" from the decoder, which is then further processed; in the locally consistent model it would be impossible to construct this codeword.

The work of [KN22]. [KN22] shows how to watermark PRFs in a quantum decoder model. Most relevant to us, their algorithm surprisingly extracts more than a logarithmic amount of information. This is because, in their setting, they essentially show that each measurement does not alter the decoder, so they can effectively work in the Globally Consistent model, despite the decoder being quantum.

At first glance, this may seem impossible: if a measurement has any chance of revealing useful information, it would seem it must also risk disturbing the quantum state; this is the famous measurement principle. However, the clever observation of [KN22] is that, in their setting, the success probably in any query will be a fixed deterministic function of the query and also the message embedded in the watermark. As such, by the gentle measurement principle, measuring the success probability will not alter the quantum state.

The problem with this approach is that it seems inherently limited to the case where there is a single marked program, and it does not seem it can handle even the case where the adversary gets two marked programs. Likewise, we expect their results could naturally be adapted to give embedded identity traitor tracing secure against a single user, but will not allow for any collusions amongst users. The reason is that each user will have a different function mapping queries to success probabilities (this is necessary to allow for tracing). But an adversary with multiple programs/users can create a decoder which sometimes uses one program, sometimes another, and which program used could be controlled by some quantum state. For such a decoder, it is no longer the case that the success probability deterministically depends on the embedded messages, and measuring success probabilities will necessarily alter the decoder. Therefore, a different approach is necessary.

2.3 Our Solution: The Back One Step (BOS) Model

A starting point for our results is the recent work of Chiesa et al. [CMSZ21]. They provide a powerful new rewinding technique, where they show that in some sense it is possible to "repair" a quantum state after a measurement. They apply their technique to the setting of interactive proofs, where they repair the quantum state of the adversary in order to rewind it many times. In this work, we show how to adapt the rewinding technique to our setting.

Before getting into the details of our approach, we highlight some conceptual differences between our settings:

- Chiesa et al. extract information from an adversary using the adversary's actual messages; the rewinds are used to obtain many such messages. The inputs for the different rewinds are typically all uniformly random and independent. In contrast, for traitor tracing, information is generally extracted by looking at success probabilities. The distributions tested will generally be far from uniform, in order to get differences in success probabilities.
- The decision tree in traitor tracing is rather deep. Consider for example the binary search algorithm for PLBE with large identities. In order to find a gap, we must explore a deep branch. We may not find anything, in which case we must return to the root and explore a different branch. In contrast, the rewinding in Chiesa et al. always returns the adversary to a state approximating the initial state of the adversary. In this sense, the decision tree is shallow, having only one level.
- The repair procedure can only repair a single feature of the adversary. This is inherent, since different quantities may correspond to "incompatible" observables, and in general quantum systems one cannot simultaneously "know" the values of incompatible observables. Chiesa et al. always repair just the original success probability of the adversary. This is enough for them, due to their shallow decision tree. Looking at our setting, however, always repairing to the initial setting will not work. For example, in the binary search case, if we just repair to the initial success probability, this allows us to return to the root of the binary search tree. But now the values at all points may have shifted around. So if we explore a branch and fail to find a gap, when we return to the root, the gap we are looking for may have moved into the branch we just abandoned. Even if we explore all branches, we may never end up actually finding the ever-moving gap.

In order to simplify the task of designing new tracing protocols to overcome these challenges, we propose an intermediate model which we call the Back One Step (BOS) model. The BOS model allows us to abstract away the complicated techniques of state repair in order to give a clean model of what the technique should be capable of. This model will then allow us to design novel tracing algorithms. Importantly, the task of designing algorithms in the model will be entirely classical, with all the quantum aspects hidden beneath the abstraction.

The Model. We now describe the BOS model: there is a collection $\mathcal{Q}$ of queries available. There is moreover a potentially *stateful* oracle which contains a secret partition Π of $\mathcal{Q}$. The oracle accepts a sequence of queries $q_1, \ldots, q_t \in \mathcal{Q}$, and answers queries with a bit $o_i \in \{0, 1\}$. Note that we discretized the o_i, to make the model simpler. We impose the following constraints:

- **Large value on source if first query.** A distinguished query $\alpha \in \mathcal{Q}$. If $q_1 = \alpha$, then $o_1 = 1$. There are no guarantees on subsequent queries to α, or if α is not the very first query.
- **Small values on sinks.** There is a collection $\Omega \subset \mathcal{Q}$, such that if $q_i \in \Omega$, then $o_i = 0$. This holds for *all* $i \in [t]$.
- **Local consistency.** For any two *consecutive* queries q_i, q_{i+1}, if q_i, q_{i+1} lie in the same part of the partition Π (and in particular, if $q_i = q_{i+1}$), then $o_i = o_{i+1}$. There is no such requirement for q_i, q_j with $|i - j| > 1$.

- **Single step rewinding.** Finally, if $q_{i+2} = q_i$, then $o_{i+2} = o_i$. Any other identical queries that may exist in the query sequence have no such restriction.

If we ignore the last requirement, note that we recover the model of Zhandry. The last requirement essentially says that we can make a query q_i, make an arbitrary different query q_{i+1}, and then rewind to the previous query q_i for query $i+2$ and be guaranteed to recover the same output o_i a second time.

Realizing the BOS Model. By adapting the rewinding technique of Chiesa et al., we show how to instantiate a BOS oracle from any pirate decoder or diO adversary. The rough idea is to run the BOS algorithm, using Zhandry's [Zha20] success probability estimation routine to answer every query. This already ensures large values on the source (if the first query), small values on the sinks, and local consistency, following the same analysis as Zhandry. In order to ensure single step rewinding, we then add one more feature: if any query q_{i+2} is equal to q_i, instead of naively computing the success probability o_{q+2} using Zhandry, we instead use the quantum rewinding technique of Chiesa et al. to rewind to a state where query $q_{i+2} = q_i$ yields success probability essentially equal to o_q. Thus, we can answer the query with $o_{q+2} = o_q$, thereby guaranteeing single step rewinding.

Designing Algorithms for the BOS Model. The BOS oracle guarantees are much weaker than the Globally Consistent Hidden Partition Model, and as discussed above, many classical tracing algorithm crucially rely on these stronger consistency guarantees. Perhaps surprisingly, while local consistency alone does not appear enough for many tracing settings, the ability to rewind even a single step opens many doors. We design new algorithms for the BOS model for the various settings, crucially leveraging the ability to rewind a single step. Importantly, this design process is entirely classical, greatly simplifying the design task. We note that the algorithm for each result is different (aside from the iO/diO equivalence and basic embedded identity tracing scheme being the same).

We next briefly describe how the process works for tracing PLBE; for the other results see the main body of the paper.

The case of PLBE. As explained above, in the case of PLBE, we have $\mathcal{Q} = [0, N], \alpha = N, \Omega = \{0\}$, and $\Pi = \{[0, a_1), [a_1, a_2), [a_2, a_3), \cdots, [a_t, N]\}$. Our goal is to recover one of the a_i.

We know if the first query is on $\alpha = N$, then the response is $o_N = 1$. Likewise, we know if we ever query on 0, the response is $o_0 = 0$. In the classical case, we would identify an a_i via a binary search. We first assign a pivot to be $N/2$, and query on $N/2$. If the query response is $o_{N/2} = 1$, we know that at least one a_i lies in the interval $[1, N/2]$, so we can recurse on the interval $[0, N/2]$. If the query response is $o_{N/2} = 0$, then we know at least one a_i lies in the interval $[N/2+1, N]$, and so we can recurse on the interval $[N/2, N]$.

In the BOS model, we can try the same: after querying on N and receiving response 1, we query on $N/2$ and receive response $o_{N/2}$. If $o_{N/2} = 1$, we know by the rules of the BOS model that there is an a_i in the interval $[1, N/2]$, and if $o_{N/2} = 0$, we know by the rules of the BOS model that there is an a_i in the interval $[N/2+1, N]$. Moreover, in the case $o_{N/2} = 1$, we actually have the correct setup to recurse on $\mathcal{Q}' = [0, N/2]$ by setting $\alpha' = N/2$, $\Omega' = \{0\}$, and Π' to be the partition of $\mathcal{Q}'$ induced by $\mathcal{Q}$. Indeed, by local consistency, since the last query was on $N/2 = \alpha'$ and the response was 1, if the next query is on α' the response will be 1, guaranteeing a large response for the recursion. Small value on the sink, local consistency, and single step rewinding carry over as well.

On the other hand, let us see what happens if we try to recurse on $[N/2, N]$ in the case $o_{N/2} = 0$. We would naturally set $\mathcal{Q}' = [N/2, N], \alpha' = N, \Omega = \{N/2\}$. We note that local consistency and single step rewinding still carry over to this case. Moreover, since the penultimate query was on $N = \alpha'$, single step rewinding guarantees that we get a large value on α' if it is the next query.

This leaves small values on sink. Initially, it is true that querying on the sink $N/2$ gives 0, as desired. But importantly this only holds initially, where the BOS model requires the sink to give 0 *always*. We can see where this leads to trouble by taking the recursion a couple steps forward. The next step is to query on $3N/4$, receiving $o_{3N/4}$. Suppose $o_{3N/4} = 1$. Then we recurse left on the interval $\mathcal{Q}'' = [N/2, 3N/4]$ and query on $5N/8$. Suppose again that $o_{5N/8} = 1$, so we recurse on the interval $[N/2, 5N/8]$. At this point, we would want that $\Omega''' = \{N/2\}$. However, our last query on $N/2$ was three queries ago (with $3N/4$ and $5N/2$ between). Therefore, $o_{N/2}$ may have changed, and there is no longer any guarantee of having small values on the sinks. Without a small value on the sinks, the recursion will fail.

One attempt to fix this problem is, after querying on $3N/4$ and receiving $o_{3N/4} = 1$, we insert a query on $N/2$. By single step rewinding, we have, as desired, that the result is $o_{N/2} = 0$. However, this strategy fails when we move to the query on $5N/8$. Suppose now that $o_{5N/8} = 0$, meaning we recurse right on the interval $\mathcal{Q}''' = [5N/8, 3N/4]$. Unfortunately, by inserting the new query on $N/2$, the most recent query on $3N/4$ was now three queries ago (being followed by $N/2$ and $5N/8$). Therefore, we no longer have the guarantee that querying on the new source $\alpha''' = 3N/4$ gives a 1, and in fact it could be that all future queries give a 0, which clearly does not allow for extracting any information.

We now explain our solution. If $o_{N/2} = 1$, we recurse on $[0, N/2]$ as we would classically. In the case $o_{N/2} = 0$, we still query on $3N/4$ to get $o_{3N/4}$. Here, if $o_{3N/4} = 1$, we cannot recurse on the interval $[N/2, 3N/4]$. However, we *can* recurse on the interval $\mathcal{Q}' = [0, 3N/4]$, since now $\Omega' = \{0\} = \Omega$ and so we guarantee small values on sinks throughout, not just on the next query.

What if $o_{3N/4} = 0$? In this case, we update the pivot to $7N/8$ as if we were recursing to the right. Importantly, however, we always keep the bottom of the domain as 0 to ensure small values on the sink. So if $o_{7N/8}$ gives 1, we recurse on the interval $[0, 7N/8]$. If $o_{7N/8} = 0$, we update the pivot to $15N/16$, and so on.

With this new algorithm, we can prove that we do indeed preserve all the properties of the BOS model in each step of the recursion, and will eventually reach the case where the interval is $[0, a]$ and the pivot is $a - 1$, and the last two queries were on $a, a - 1$ giving $o_a = 1, o_{a-1} = 0$. We output a, as local consistency implies that a is one of the endpoints of the intervals in Π. Moreover, we can prove termination in a polynomial number of steps, namely $O(t \log^2 N)$ where t is the number of a_i.

3 Quantum Background

3.1 Basic Quantum Preliminaries

Much of this section is taken almost verbatim from [CMSZ21]. A (pure) *quantum state* is a vector $|\psi\rangle$ in a complex Hilbert space $\mathcal{H}$ with $\||\psi\rangle\| = 1$; in this work, $\mathcal{H}$ is finite-dimensional. We denote by $\mathbf{S}(\mathcal{H})$ the space of Hermitian operators on $\mathcal{H}$. A *density matrix* is a positive semi-definite operator $\boldsymbol{\rho} \in \mathbf{S}(\mathcal{H})$ with $\mathsf{Tr}(\boldsymbol{\rho}) = 1$. A density matrix represents a probabilistic mixture of pure states (a mixed state); the density matrix corresponding to the pure state $|\psi\rangle$ is $|\psi\rangle\langle\psi|$. Typically we divide a Hilbert space into *registers*, e.g. $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$. We sometimes write, e.g., $\boldsymbol{\rho}^{\mathcal{H}_1}$ to specify that

$\boldsymbol{\rho} \in \mathbf{S}(\mathcal{H}_1)$.

A unitary operation is a complex square matrix U such that $UU^\dagger = \mathbf{I}$. The operation U transforms the pure state $|\psi\rangle$ to the pure state $U|\psi\rangle$, and the density matrix $\boldsymbol{\rho}$ to the density matrix $U\boldsymbol{\rho}U^\dagger$.

A *projector* P is a Hermitian operator ($P^\dagger = P$) such that $P^2 = P$. A *projective measurement* is a collection of projectors $\mathbf{P} = (P_i)_{i \in S}$ such that $\sum_{i \in S} P_i = \mathbf{I}$. This implies that $P_i P_j = 0$ for distinct i and j in S. The application of $\mathbf{P}$ to a pure state $|\psi\rangle$ yields outcome $i \in S$ with probability $p_i = \|P_i|\psi\rangle\|^2$; in this case the post-measurement state is $|\psi_i\rangle = P_i|\psi\rangle/\sqrt{p_i}$. We refer to the post-measurement state $|\psi_i\rangle$ as the result of applying $\mathbf{P}$ to $|\psi\rangle$ and *post-selecting* (conditioning) on outcome i. A state $|\psi\rangle$ is an *eigenstate* of $\mathbf{P}$ if it is an eigenstate of every P_i.

A two-outcome projective measurement is called a *binary projective measurement*, and is written as $\mathbf{P} = (P, \mathbf{I} - P)$, where P is associated with the outcome 1, and $\mathbf{I} - P$ with the outcome 0.

General (non-unitary) evolution of a quantum state can be represented via a *completely-positive trace-preserving (CPTP)* map $T\colon \mathbf{S}(\mathcal{H}) \to \mathbf{S}(\mathcal{H}')$. We omit the precise definition of these maps in this work; we only use the facts that they are trace-preserving (for every $\boldsymbol{\rho} \in \mathbf{S}(\mathcal{H})$ it holds that $\mathsf{Tr}(T(\boldsymbol{\rho})) = \mathsf{Tr}(\boldsymbol{\rho})$) and linear.

For Hilbert spaces $\mathcal{A}, \mathcal{B}$ the *partial trace* over $\mathcal{B}$ is the unique CPTP map $\mathsf{Tr}_{\mathcal{B}}\colon \mathbf{S}(\mathcal{A} \otimes \mathcal{B}) \to \mathbf{H}(\mathcal{A})$ such that $\mathsf{Tr}_{\mathcal{B}}(\boldsymbol{\rho}_A \otimes \boldsymbol{\rho}_B) = \mathsf{Tr}(\boldsymbol{\rho}_B)\boldsymbol{\rho}_A$ for every $\boldsymbol{\rho}_A \in \mathbf{S}(\mathcal{A})$ and $\boldsymbol{\rho}_B \in \mathbf{S}(\mathcal{B})$.

For every CPTP map $T\colon \mathbf{S}(\mathcal{H}) \to \mathbf{S}(\mathcal{H})$there exists a *unitary dilation* U that operates on an expanded Hilbert space $\mathcal{H} \otimes \mathcal{K}$, so that $T(\boldsymbol{\rho}) = \mathsf{Tr}_{\mathcal{K}}(U(\boldsymbol{\rho} \otimes |0\rangle\langle 0|^{\mathcal{K}})U^\dagger)$. This is not necessarily unique; however, if T is described as a circuit then there is a dilation U_T represented by a circuit of size $O(|T|)$.

A *general measurement* is a CPTP map $\mathbf{M}\colon \mathbf{S}(\mathcal{H}) \to \mathbf{S}(\mathcal{H} \otimes \mathcal{O})$, where $\mathcal{O}$ is an ancilla register holding a classical outcome. Specifically, given measurement operators $\{M_i\}_{i=1}^N$ such that $\sum_{i=1}^N M_i M_i^\dagger = \mathbf{I}$ and a basis $\{|i\rangle\}_{i=1}^N$ for $\mathcal{O}$, $\mathbf{M}(\boldsymbol{\rho}) = \sum_{i=1}^N (M_i \boldsymbol{\rho} M_i^\dagger \otimes |i\rangle\langle i|^{\mathcal{O}})$. We sometimes implicitly discard the outcome register. A projective measurement is a general measurement where the M_i are projectors. A measurement induces a probability distribution over its outcomes given by $\Pr[i] = \mathsf{Tr}(|i\rangle\langle i|^{\mathcal{O}}\mathbf{M}(\boldsymbol{\rho}))$; we denote sampling from this distribution by $i \leftarrow \mathbf{M}(\boldsymbol{\rho})$.

The *trace distance* between states $\boldsymbol{\rho}, \boldsymbol{\rho}'$, denoted $d(\boldsymbol{\rho}, \boldsymbol{\rho}')$, and is defined as $\frac{1}{2}\mathsf{Tr}(\sqrt{(\boldsymbol{\rho} - \boldsymbol{\rho}')^2})$. The trace distance is contractive under CPTP maps (for any CPTP map T, $d(T(\boldsymbol{\rho}), T(\boldsymbol{\rho}')) \leq d(\boldsymbol{\rho}, \boldsymbol{\rho}')$). It follows that for any measurement $\mathbf{M}$, the statistical distance between the distributions $\mathbf{M}(\boldsymbol{\rho})$ and $\mathbf{M}(\boldsymbol{\rho}')$ is bounded by $d(\boldsymbol{\rho}, \boldsymbol{\rho}')$.

Quantum algorithms. In this work, a *quantum algorithm* is a quantum circuit built from some universal quantum gate set. We will usually consider the Hilbert space for a quantum algorithm as the product of of three registers $\mathcal{H}_{\mathsf{in}} \times \mathcal{H}_{\mathsf{out}} \times \mathcal{H}_{\mathsf{work}}$. Here, input x is mapped to $|x\rangle \in \mathcal{H}_{\mathsf{in}}$ where $\{|x\rangle\}$ is some basis for $\mathcal{H}_{\mathsf{in}}$, the output of the algorithm is obtained by measuring $\mathcal{H}_{\mathsf{out}}$ in some basis, and $\mathcal{H}_{\mathsf{work}}$ is an incilla register used as work space. A quantum polynomial time (QPT) algorithm is one where the size of the quantum circuit is polynomial.

We will also consider *quantum state algorithms*. This is a circuit $\mathcal{A}$ acting on registers $\mathcal{H}_{\mathsf{in}} \times \mathcal{H}_{\mathsf{out}} \times \mathcal{H}_{\mathsf{work}} \times \mathcal{H}_{\mathsf{aux}}$ together together with a state $|\psi\rangle \in \mathbf{S}(\mathcal{H}_{\mathsf{aux}})$. A quantum state algorithm works just as a quantum algorithm, except that $\mathcal{H}_{\mathsf{aux}}$ is initialized to $|\psi\rangle$ before applying $\mathcal{A}$. We will often denote quantum state algorithms by $|A\rangle$, which will be interpreted as including both $\mathcal{A}$ and $|\psi\rangle$.

Black-box access A circuit C with black-box access to a unitary U, denoted C^U, is a standard quantum circuit with special gates that act as U and $U^\dagger$. We also use C^T to denote black-box access to a map T, which we interpret as C^{U_T} for a unitary dilation U_T of T; all of our results are independent of the choice of dilation. This allows, for example, the "partial application" of a projective measurement, and the implementation of a general measurement via a projective measurement on a larger space.

3.2 Useful definitions and Lemmas

We distinguish between classical probabilistic polynomial time (PPT) algorithms and quantum polynomial time (QPT) algorithms. Sometimes we will consider programs that consist of a quantum state. Formally, these will consist of a quantum circuit C, and and a quantum auxiliary input aux. To evaluate the program on input x, one evaluates C on $|x\rangle \otimes \mathsf{aux}$. We will denote such programs by $|P\rangle$ and the evaluation of such programs as $|P\rangle(x)$.

Almost Projective Measurements. We state a property of general measurements due to [Zha20] that captures when a measurement is "close" to being projective, in the sense that sequential applications of the measurement yield similar outcomes.

Definition 3.1. A real-valued measurement $\mathbf{M} = (M_p)_p$ is (ϵ, δ)-*almost projective* if applying $\mathbf{M}$ twice in a row to a register $\mathcal{H}$ (initially containing any state $\boldsymbol{\rho}$) produces measurement outcomes p, p' where $\Pr[|p - p'| \leq \epsilon] \geq 1 - \delta$.

Quantum State Repair. We now recall quantum state repair from [CMSZ21].

Lemma 3.2 ([CMSZ21], Lemma 4.10)**.** *Given a projective measurement $\mathbf{P}$ on register $\mathcal{H}$ that has outcomes in set S of size N, an (ϵ, δ)-almost projective measurement $\mathbf{M}$ on $\mathcal{H}$, and $T \in \mathbb{N}, s \in S, p \in [0,1]$, there exists a procedure $\mathsf{Repair}^{\mathbf{M},\mathbf{P}}_{T,p,s}$ on $\mathcal{H}$ such that:*

- *(State is repaired with respect to $\mathbf{M}$) Consider applying the following operations to register $\mathcal{H}$ initially containing state $\boldsymbol{\rho}$:*
  1. *First apply $\mathbf{M}$ to obtain $p \in [0,1]$,*
  2. *Then apply $\mathbf{P}$ to obtain outcome $s \in S$,*
  3. *Then apply $\mathsf{Repair}^{\mathbf{M},\mathbf{P}}_{T,p,s}$,*
  4. *And finally, apply $\mathbf{M}$ once more to obtain $p' \in [0,1]$.*

  Then $\Pr[|p - p'| > 2\epsilon] \leq N\delta + N/T + 4\sqrt{\delta}$.
- *(Efficiency) The expected total number of calls that Repair makes to $\mathbf{M}$ and $\mathbf{P}$ is at most $N + 4T\sqrt{\delta}$.*

In other words, since $\mathbf{M}$ is almost projective, applying $\mathbf{M}$ twice in a row will give outcomes p, p' that are close. However, if $\mathbf{P}$ is applied in between these two measurements, there are no more guarantees on the closeness of p, p'. However, by applying Repair before the second application of $\mathbf{M}$, we can once again ensure closeness of p, p'.

Mixtures of Projective Measurements. The following is taken from [Zha20]. We consider the following abstract setup. We have a collection $\mathcal{P} = \{\mathcal{P}_i\}_{i\in\mathcal{I}}$ of binary outcome projective measurements $\mathcal{P}_i = (P_i, Q_i)$ over the same Hilbert space $\mathcal{H}$. Here, P_i corresponds to output 0, and Q_i corresponds to output 1. We will assume we can efficiently measure the $\mathcal{P}_i$ for superpositions of i, meaning we can efficiently perform the following projective measurement over $\mathcal{I}\otimes\mathcal{H}$:

$$\left(\sum_i |i\rangle\langle i|\otimes P_i\ ,\ \sum_i |i\rangle\langle i|\otimes Q_i\right) \tag{1}$$

Here, we call $\mathcal{P}$ a *collection of projective measurements*, and call $\mathcal{I}$ the *control*. For a distribution D over $\mathcal{I}$, let $\mathcal{P}_D$ be the POVM which samples a random $i \leftarrow D$, applies the measurement $\mathcal{P}_i$, and outputs the resulting bit. We call $\mathcal{P}_D$ a *mixture of projective measurements*. The POVM is given by the matrices (P_D, Q_D) where

$$P = \sum_{i\in\mathcal{I}} \Pr[i\leftarrow D]P_i \quad \text{and} \quad Q = \sum_{i\in\mathcal{I}} \Pr[i\leftarrow D]Q_i$$

Next, for $a\in\mathbb{R}$ and interval $[b,c]\subseteq\mathbb{R}$, denote the distance between a and $[b,c]$ as $|a-[b,c]| := \min_{x\in[b,c]}|a-x|$. For $a\in[b,c]$, the distance is 0 and for $a\notin[b,c]$, the distance is $\max(a-c, b-a)$. Let D_0, D_1 be two distributions over $\mathbb{R}$, with cumulative density functions f_0, f_1, respectively. Let $\epsilon\in\mathbb{R}$. The Shift distance with parameter ϵ is defined as:

$$\Delta_\epsilon(D_0, D_1) := \sup_{x\in\mathbb{R}} \left|f_0(x) - [f_1(x-\epsilon), f_1(x+\epsilon)]\right|$$

Let $\mathbf{M} = (M_i)_{i\in\mathcal{I}}$ and $\mathbf{N} = (N_j)_{j\in\mathcal{J}}$ be real-valued quantum measurements over the same quantum system $\mathcal{H}$. The shift distance between $\mathbf{M}, \mathbf{N}$, denoted $\Delta_\epsilon(\mathbf{M}, \mathbf{N})$ is defined as

$$\Delta_\epsilon(\mathbf{M}, \mathbf{N}) := \sup_{|\psi\rangle} \Delta_\epsilon(\ \mathbf{M}(|\psi\rangle)\ ,\ \mathbf{N}(|\psi\rangle)\ )$$

Now, if $\mathcal{P}_D = (P_D, Q_D)$ is a mixture of projective measurements, we note that $Q_D = \mathbf{I} - P_D$, and therefore P_D, Q_D commute. In this case, $\mathcal{P}_D$ has a *projective implementation*, denoted $\mathsf{ProjImp}(\mathcal{P}_D)$, which is defined as follows. Let S be the set of eigenvalues of P_D, and R_i for i the projectors onto the associated eigenspaces. Then $\mathsf{ProjImp}(\mathcal{P}_D)$ is the projective measurement $(P_i)_{i\in S}$. Note that $S\subseteq[0,1]$. Also note that applying $\mathcal{P}_D$ is equivalent to the following: first apply $\mathsf{ProjImp}(\mathcal{P}_D)$ to obtain outcome p, then interpret p as a probability and output 1 with probability p.

Lemma 3.3 ([Zha20], Theorem 6.2)**.** *For any $\epsilon, \delta, \mathcal{P}, D$, there exists an algorithm $\mathsf{API}_{\epsilon,\delta}^{\mathcal{P},D}$ operating on $\mathcal{H}$ and making quantum queries to $\mathcal{P}, D$ which additionally outputs a number in some set $S\subseteq[0,1]$ such that:*

- *There is a function $R = \mathsf{poly}(1/\epsilon, \log(1/\delta))$ such that the expected number of calls $\mathsf{API}_{\epsilon,\delta}^{\mathcal{P},D}$ makes to $\mathcal{P}$ and D, the running time of $\mathsf{API}_{\epsilon,\delta}^{\mathcal{P},D}$, and $|S|$ are all bounded by R.*
- *$\mathsf{API}_{\epsilon,\delta}^{\mathcal{P},D}$ is (ϵ,δ)-almost projective.*
- *$\Delta_\epsilon(\mathsf{API}_{\epsilon,\delta}^{\mathcal{P},D}, \mathsf{ProjImp}(\mathcal{P}_D)) \le \delta$.*

Lemma 3.4 ([Zha20], Theorem 6.5)**.** *Let $\boldsymbol{\rho}$ be an efficiently constructible mixed state over $\mathcal{H}$, and D_0, D_1 efficiently sampleable, computationally indistinguishable distributions. For any inverse polynomial ϵ, there exists a negligible δ such that $\Delta_\epsilon(\mathsf{ProjImp}(\mathcal{P}_{D_0})(\boldsymbol{\rho}), \mathsf{ProjImp}(\mathcal{P}_{D_0})(\boldsymbol{\rho})) \le \delta$.*

4 Cryptographic Notions

Note that we use superscripts like $^{\mathsf{IPP}}$ and $^{\mathsf{TT}}$ to disambiguate between algorithms belonging to different cryptosystems.

4.1 Functional Encryption

A functional encryption scheme is a tuple $\Pi^{\mathsf{FE}} = (\mathsf{Gen}^{\mathsf{FE}}, \mathsf{Derive}^{\mathsf{FE}}, \mathsf{Enc}^{\mathsf{FE}}, \mathsf{Dec}^{\mathsf{FE}})$ defined as follows:

- $\mathsf{Gen}^{\mathsf{FE}}(1^\lambda)$ is a classical probabilistic polynomial time (PPT) algorithm that takes as input the security parameter λ in unary, and samples a public key pk and master secret key msk.
- $\mathsf{Derive}^{\mathsf{FE}}(\mathsf{msk}, f)$ is a classical PPT algorithm that takes as input the master secret key msk and a classical circuit f. It outputs a secret key sk_f.
- $\mathsf{Enc}^{\mathsf{FE}}(\mathsf{pk}, m)$ is a classical PPT algorithm that takes as input the public key pk and a message m, and produces a ciphertext c.
- $\mathsf{Dec}^{\mathsf{FE}}(\mathsf{sk}_f, c)$ is a classical deterministic algorithm that takes as input a secret key sk_f for a function f and a ciphertext, and outputs a value x.

Definition 4.1. A functional encryption scheme Π^{FE} is *correct* if, for all messages m and functions f,

$$\Pr\left[\mathsf{Dec}^{\mathsf{FE}}(\mathsf{sk}_f, \mathsf{Enc}^{\mathsf{FE}}(\mathsf{pk}, m)) = f(m) : \begin{smallmatrix}(\mathsf{pk},\mathsf{msk})\leftarrow\mathsf{Gen}^{\mathsf{FE}}(1^\lambda)\\ \mathsf{sk}_f\leftarrow\mathsf{Derive}^{\mathsf{FE}}(\mathsf{msk},f)\end{smallmatrix}\right] \geq 1 - \mathsf{negl}(\lambda)$$

We now discuss security. For an adversary $\mathcal{A}^{\mathsf{FE}}$, consider the following experiment on $\mathcal{A}^{\mathsf{FE}}$:

- Run $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Gen}^{\mathsf{FE}}(1^\lambda)$, and send pk to $\mathcal{A}^{\mathsf{FE}}$.
- $\mathcal{A}^{\mathsf{FE}}$ can now make an arbitrary number of classical secret key queries on functions f; in response it receives sk_f.
- At some point, $\mathcal{A}^{\mathsf{FE}}$ makes a single challenge query on a pair of messages m_0^*, m_1^*. The challenger chooses a random bit b, and sends $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b^*)$.
- $\mathcal{A}^{\mathsf{FE}}$ can continue making secret key queries. Let F be the set of all f queried by $\mathcal{A}^{\mathsf{FE}}$.
- Finally, $\mathcal{A}^{\mathsf{FE}}$ makes a guess b' for b.

Definition 4.2. A functional encryption scheme Π^{FE} is secure if, for all QPT $\mathcal{A}^{\mathsf{FE}}$, there exists a negligible negl such that:

$$\Pr\left[b' = b \wedge f(m_0^*) = f(m_1^*) \forall f \in F\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Variations. We can consider a number of variations. We could insist on a secret key scheme, where there is no pk and msk is needed to encrypt. We could consider the case of bounded collusions, where an additional input n is given to $\mathsf{Gen}^{\mathsf{FE}}$, and the security game only allows up to n secret key queries. We could also limit the class of functions, rather than considering general efficiently computable functions. The case of ordinary public key encryption is that where the class of functions is empty (and so the procedure $\mathsf{Derive}^{\mathsf{FE}}$ is vacuous). In this case, we denote the protocol by $\Pi^{\mathsf{PK}} = (\mathsf{Gen}^{\mathsf{PK}}, \mathsf{Enc}^{\mathsf{PK}}, \mathsf{Dec}^{\mathsf{PK}})$.

4.2 Indistinguishability Obfuscation

Indistinguishability obfuscation (iO) is defined by [BGI+01]. We utilize a uniform notion, which was defined in [GGH+13], and which we adapt here to the quantum setting.

Definition 4.3 (Indistinguishability Obfuscation)**.** A post-quantum *indistinguishability obfuscator* (iO) is a PPT algorithm iO such that:

- $\mathsf{iO}(1^\lambda, C)$ is equivalent to C with overwhelming probability over the randomness of iO.
- Let $\mathcal{S}(1^\lambda)$ be a quantum algorithm sampling two classical circuits C_0, C_1 of the same size, as well as a quantum auxiliary input aux. We say $\mathcal{S}$ is *valid* if, with overwhelming probability in λ, C_0, C_1 are equivalent. Security requires, for any efficient valid $\mathcal{S}$, the distributions $(\mathsf{iO}(1^\lambda, C_0), C_0, C_1, \mathsf{aux})$ and $(\mathsf{iO}(1^\lambda, C_1), C_0, C_1, \mathsf{aux})$ are computationally indistinguishable.

4.3 IPP Codes

A fingerprinting code [BS95] is used to fingerprint data. Fingerprinting codes have long been used to build traitor tracing schemes (e.g. [BN08, Sir06, BP08]). Note, however, that the fingerprinting codes explicitly cited in the literature are usually *binary* codes. Yet, the schemes provided can be generalized to non-binary codes, but require a different kind of code called an identifiable parent property (IPP) code [HvLT98]. We will use this generalization in order to abstract more schemes from the literature, so we here define IPP codes rather than fingerprinting codes. Note that the two notions coincide for binary codes.

An IPP code is a pair $(\mathsf{Gen}^{\mathsf{IPP}}, \mathsf{Trace}^{\mathsf{IPP}})$ of PPT algorithms such that:

- $\mathsf{Gen}^{\mathsf{IPP}}(1^\lambda, 1^N, 1^c)$ takes as input a security parameter, a number of users N, and a collusion bound c, all in unary. It outputs a tracing key tk, as well as N strings $w_1, \ldots, w_N \in \Sigma^\ell$. The w_i are called codewords
- $\mathsf{Trace}^{\mathsf{IPP}}(\mathsf{tk}, w^*)$ takes as input the tracing key and a codeword $w^* \in \Sigma^\ell$. It outputs a set $A \subseteq [N]$.

For a set of codewords $C \subseteq \Sigma^\ell$, let $F(C)$ be the "feasible set" of C, which is defined as follows. A word $w^* \in \Sigma \cup \{?\}$ is in $F(C)$ if, for each position $j \in [\ell]$, either $w^*_j =?$ or there exists a codeword $w \in C$ such that $w^*_j = w_j$. In other words, $F(C)$ consists of all codewords in C, plus all codewords that can be obtained from C by potentially using a different codeword character in each position. Additionally, any entry can be set to ?. The condition of being in the feasible set of C is also called the "marking condition".

Remark 4.4. Where IPP codes differ from fingerprinting codes is in this marking condition. For a fingerprinting code, in any position j where $\{w_j\}_{w \in C}$ is not a single character, w^*_j is allowed to be *anything*. This condition better reflects the use in fingerprinting. However, the marking condition for IPP codes better reflects the use in traitor tracing.

Definition 4.5. An IPP code $(\mathsf{Gen}^{\mathsf{IPP}}, \mathsf{Trace}^{\mathsf{IPP}})$ is δ-robust if, for all (potentially unbounded) algorithms $\mathcal{A}^{\mathsf{IPP}}$, for all polynomially bounded N, c, and for all $S \subseteq [N]$, $\mathcal{A}^{\mathsf{IPP}}$ wins the following experiment with probability at most $2^{-\lambda}$:

- Run $(\mathsf{tk}, \{w_i\}_{i \in [N]}) \leftarrow \mathsf{Gen}^{\mathsf{IPP}}(1^\lambda, 1^N, 1^c)$.

- Run $w^* \leftarrow \mathcal{A}^{\mathsf{IPP}}(\{w_i\}_{i\in S})$
- If the number of ? in w^* is more than $\delta\ell$, output "lose" and stop.
- Otherwise, run $A \leftarrow \mathsf{Trace}^{\mathsf{IPP}}(\mathsf{tk}, w^*)$
- Output "win" if (1) $w^* \in F(\{w_i\}_{i\in S})$ (w^* is feasible for the set of codewords given to $\mathcal{A}^{\mathsf{IPP}}$), and (2) either $A = \emptyset$ or $A \not\subseteq S$.

Instantiations. In the case of binary alphabets, δ-robust IPP codes are also fingerprinting codes. Optimal fingerprinting codes with constant δ have $\ell = \Theta(c^2\lambda^2)$ [Tar03]. In the case of non-binary alphabets, less is known about IPP codes. However, the main traitor tracing scheme of [CFN94] can be seen as employing IPP codes with $\delta = 0$ and various parameters choices for ℓ, Σ, including $\ell = \Theta(c\lambda), |\Sigma| = \Theta(c^2\lambda)$.

4.4 Traitor Tracing with Quantum Decoders

We follow the definitions of traitor tracing given by Zhandry [Zha20]. The text is mostly taken from Zhandry, except we adjust the syntax to accommodate both ordinary and embedded-identity traitor tracing. A traitor tracing system for identity space $\mathcal{I}$ is a tuple $\Pi^{\mathsf{TT}} = (\mathsf{Gen}^{\mathsf{TT}}, \mathsf{Derive}^{\mathsf{TT}}, \mathsf{Enc}^{\mathsf{TT}}, \mathsf{Dec}^{\mathsf{TT}}, \mathsf{Trace}^{\mathsf{TT}})$ defined as follows:

- $\mathsf{Gen}^{\mathsf{TT}}(1^\lambda)$ is a classical probabilistic polynomial time (PPT) algorithm that takes as input the security parameter (in unary), and samples a public key pk and master secret key msk.
- $\mathsf{Derive}^{\mathsf{TT}}(\mathsf{msk}, \mathsf{id})$ is a classical PPT algorithm that takes as input the master secret key msk and an identity $\mathsf{id} \in \mathcal{I}$, and outputs a user secret key $\mathsf{sk}_{\mathsf{id}}$.
- $\mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m)$ is a classical PPT algorithm that takes as input the public key pk and a message m, and outputs a ciphertext c.
- $\mathsf{Dec}^{\mathsf{TT}}(\mathsf{sk}_{\mathsf{id}}, c)$ is a classical deterministic algorithm that takes as input a secret key $\mathsf{sk}_{\mathsf{id}}$ for user id and a ciphertext, and outputs a message m'.
- $\mathsf{Trace}^{\mathsf{TT}}(\mathsf{pk}, m_0, m_1, 1^{1/\epsilon}, |\mathsf{D}\rangle)$ is a QPT algorithm that takes as input the public key pk, two messages m_0, m_1, and a parameter ϵ (whose reciprocal is an integer represented in unary), and a quantum state $|\mathsf{D}\rangle$ representing a pirate decoder. It ultimately outputs a subset of $\mathcal{I}$, which are the accused users. Here, m_0, m_1 are two messages whose ciphertexts $|\mathsf{D}\rangle$ supposedly distinguishes, and ϵ is a supposed lower bound on the distinguishing advantage.

We next define correctness and security.

Definition 4.6. A traitor tracing system Π^{TT} is *correct* if, for all messages m and identities $\mathsf{id} \in \mathcal{I}$,

$$\Pr\left[\mathsf{Dec}^{\mathsf{TT}}(\mathsf{sk}_{\mathsf{id}}, \mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m)) = m : \begin{smallmatrix}(\mathsf{pk},\mathsf{msk})\leftarrow\mathsf{Gen}^{\mathsf{TT}}(1^\lambda)\\ \mathsf{sk}_{\mathsf{id}}\leftarrow\mathsf{Derive}^{\mathsf{TT}}(\mathsf{msk},\mathsf{id})\end{smallmatrix}\right] \geq 1 - \mathsf{negl}(\lambda)$$

We now discuss security, adapting the software decoder model version of the definition of Zhandry [Zha20]. For a decoder $|\mathsf{D}\rangle$, two messages m_0, m_1, and public key pk, consider the operation on $|\mathsf{D}\rangle$:

- Choose a random bit $b \leftarrow \{0,1\}$.
- Run $c \leftarrow \mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m_b)$ to get a random encryption of m_b.
- Run $b' \leftarrow |\mathsf{D}\rangle(c)$.
- Output 1 if and only if $b = b'$; otherwise output 0.

Let $\mathcal{M}^{\mathsf{TT}} = (M_0, M_1)$ be the POVM given by this operation, which we call the *associated POVM* to the decoder. $\mathcal{M}^{\mathsf{TT}}$ has a projective implementation $\mathsf{ProjImp}(\mathcal{M}^{\mathsf{TT}}) = \{M'_p\}_p$, where each M'_p corresponds to the probability distribution on $\{0,1\}$ that is 1 with probability p.

Tracing Experiment. For an adversary $\mathcal{A}^{\mathsf{TT}}$, function $\epsilon(\cdot)$, and security parameter λ, we consider the following experiment on $\mathcal{A}^{\mathsf{TT}}$:

- Run $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Gen}^{\mathsf{TT}}(1^\lambda)$, and send pk to $\mathcal{A}^{\mathsf{TT}}$.
- $\mathcal{A}^{\mathsf{TT}}$ then makes an arbitrary number of classical queries on identities $\mathsf{id} \in \mathcal{I}$; in response it receives $\mathsf{sk}_{\mathsf{id}}$. Let S be the set of id queried by $\mathcal{A}^{\mathsf{TT}}$.
- Next, $\mathcal{A}^{\mathsf{TT}}$ outputs $(|\mathsf{D}\rangle, m_0, m_1)$ for decoder $|\mathsf{D}\rangle$ and messages m_0, m_1.
- Apply the measurement $\mathsf{ProjImp}(\mathcal{M}^{\mathsf{TT}})$ to $|\mathsf{D}\rangle$, obtaining a probability p. Let $\mathsf{Live}^{\mathsf{TT}}_\epsilon$ be the event that $p \geq 1/2 + \epsilon$.
- Finally run $A \leftarrow \mathsf{Trace}^{\mathsf{TT}}(\mathsf{pk}, m_0, m_1, 1^{1/\epsilon}, |\mathsf{D}\rangle)$ to get a set of accused users. Let $\mathsf{Fail}^{\mathsf{TT}}_\epsilon$ as the event that $A \not\subseteq S$ (an accused user was not one of the queried users). We define the event $\mathsf{Success}^{\mathsf{TT}}_\epsilon$ as the event that $A \neq \emptyset$ (some user is accused).

Definition 4.7. A tracing system is *quantum traceable* if for all quantum polynomial time adversaries $\mathcal{A}^{\mathsf{TT}}$ and for every inverse polynomial ϵ, there is a negligible negl such that $\Pr[\mathsf{Fail}^{\mathsf{TT}}_\epsilon] < \mathsf{negl}(\lambda)$ and $\Pr[\mathsf{Live}^{\mathsf{TT}}_\epsilon \wedge \neg\mathsf{Success}^{\mathsf{TT}}_\epsilon] < \mathsf{negl}(\lambda)$.

Variations. The usual setting of traitor tracing has $\mathbf{I} = [N]$ for a polynomial N. In this case, we often set N to be an explicit input to $\mathsf{Gen}^{\mathsf{TT}}$, and the parameters of the scheme may depend on N. A bounded collusion traitor tracing scheme additionally has another parameter c given to $\mathsf{Gen}^{\mathsf{TT}}$, and security only is required to hold if $|S| \leq c$. Traitor tracing with *embedded identities* has for $\mathcal{I} = \{0,1\}^n$ for integer n. Therefore, identities are now polynomial-length strings. Here, n may be an input to $\mathsf{Gen}^{\mathsf{TT}}$ and the parameters of the scheme may depend on n. Note that some versions of embedded identity traitor tracing will have $\mathcal{I} = \{0,1\}^*$, meaning there is no a priori bound on the length of identity strings. Finally, we consider schemes with private tracing, where $\mathsf{Gen}^{\mathsf{TT}}$ outputs a special tracing key tk, which is inputted into $\mathsf{Trace}^{\mathsf{TT}}$, and security only holds if tk is kept secret.

4.5 Defining Differing Inputs Obfuscation.

We now define differing-inputs obfuscation (diO) in the quantum setting. To the best of our knowledge, diO has not been defined in the quantum setting, and it turns out that the task is somewhat subtle.

There are several definitions of diO in the classical literature, all with slight differences. For our discussion, we will focus on the distributional variants of [BCP14] and [ABG+13]. Interestingly, the existing literature largely treats these two as the same. But we observe that the two definitions are actually not equivalent. We now briefly recall the two flavors of definitions and discuss their differences.

Both definitions consider samplers $\mathcal{S}(1^\lambda)$ that sample circuits C_0, C_1 along with auxiliary input aux. The intuition for security is that if it is hard to find x such that $C_0(x) \neq C_1(x)$ (given C_0, C_1, aux), then it is hard to disitnguish obfuscations of C_0, C_1 (also given C_0, C_1, aux). Where the two definitions differ is in how this is formalized.

The Ananth et al. [ABG+13] style definition. Here, we call $\mathcal{S}$ a *differing inputs sampler* if it is computationally infeasible to, given $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{S}(1^\lambda)$, find x such that $C_0(x) \neq C_1(x)$. The definition then says that for any efficient differing inputs sampler $\mathcal{S}$, it is computationally intractable to distinguish with non-negligible advantage the following distributions:

$$(\mathsf{diO}(1^\lambda, C_0), C_0, C_1, \mathsf{aux}) \quad \text{and} \quad (\mathsf{diO}(1^\lambda, C_1), C_0, C_1, \mathsf{aux})$$

Here the two distributions are over $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{S}(1^\lambda)$ and the randomness of $\mathsf{diO}(1^\lambda, C_0), \mathsf{diO}(1^\lambda, C_1)$.

The Boyle et al. [BCP14] style definition. This definition is a more fine-grained notion. It says that for any efficient distinguisher $\mathcal{A}$ and any efficient sampler $\mathcal{S}$ (which may or may not be differing inputs), there exists an efficient extractor $\mathcal{E}$ such that the following holds with overwhelming probability over (C_0, C_1, aux): if $\mathcal{A}$ can distinguish with non-negligible advantage the distributions

$$(\mathsf{diO}(1^\lambda, C_0), C_0, C_1, \mathsf{aux}) \quad \text{and} \quad (\mathsf{diO}(1^\lambda, C_1), C_0, C_1, \mathsf{aux}) \ ,$$

then $\mathcal{E}(C_0, C_1, \mathsf{aux})$ will produce an x such that $C_0(x) = C_1(x)$. Here, importantly, the two distributions above have fixed C_0, C_1, aux, and are only over the random coins in $\mathsf{diO}(1^\lambda, C_0)$.

A subtle point is that the Boyle et al. definition only requires that $\mathcal{E}$ succeed with inverse polynomial probability whenever $\mathcal{A}$ distinguishes. One could imagine a stronger definition that requires $\mathcal{E}$ to instead succeed with probability negligibly close to 1. Fortunately, the inverse polynomial success probability of $\mathcal{E}$ is only over the random coins of $\mathcal{E}$, since the definition has already conditioned on all inputs to $\mathcal{E}$. Therefore, by simply running $\mathcal{E}$ many times, the success probability can be boosted arbitrarily high.

Either version of the Boyle et al. definition, however, turns out to be more fine grained than the Ananth et al. definition. Indeed, if $\mathcal{S}$ is a differing inputs sampler, then $\mathcal{E}$ cannot exist, and so with overwhelming probability over the choice of C_0, C_1, aux, $\mathcal{A}$ fails to distinguish the two distributions above. Averaging over all C_0, C_1, aux shows that $\mathcal{A}$ also fails to distinguish on average. In other words, the Boyle et al. definition is at least as strong as the Ananth et al. definition. The converse direction is unclear. Consider for example an $\mathcal{A}$ that has advantage on some C_0, C_1, aux, but not others. The Ananth et al. definition would imply that $\mathcal{S}$ is not differing inputs, meaning it is possible to find differing inputs for *some* C_0, C_1, aux. But the definition does not say which. It could be that the differing inputs are found for C_0, C_1, aux where $\mathcal{A}$ actually has zero advatnage, and no differing inputs are found when $\mathcal{A}$ has advantage. In contrast, the Boyle et al. definition implies that differing inputs are found exactly when $\mathcal{A}$ has advantage.

Challenges in defining diO in the quantum setting. We now turn to considering the quantum setting. We are still interested in obfuscating classical circuits using a classical obfuscation algorithm. However, we now allow adversaries to be quantum. Importantly, the auxiliary input will often be the internal state of the adversary, and so we model the sampler $\mathcal{S}$ as a quantum algorithm that produces aux that is a quantum state. In the interest of obtaining the strongest possible *positive* result, we would like the strongest definition of diO.

While diO has not been previously formalized in the quantum setting, a closely related concept of *extractable witness encryption* was considered by [GZ20] and [CLLZ21], which was defined in the flavor of [BCP14] and [ABG+13], respectively. However, the definition of [GZ20] is given as a worst-case definition of the choice of aux, rather than average case. Looking head, the iO-diO equivalence will only give an average-case diO[4]. On the other hand, the definition of [GZ20] is of the weaker variety where $\mathcal{E}$ only must succeed with inverse polynomial success probability. It turns out that arbitrarily boosting the success probability of $\mathcal{E}$ is not possible in the quantum setting: any run of $\mathcal{E}$ may "consume" the quantum auxiliary input, preventing $\mathcal{E}$ from naively being run multiple times.

In fact, the naive attempt to define security with $\mathcal{E}$ succeeding with probability almost 1 is simply impossible. This is because aux may be a superposition of two different auxiliary inputs, aux_0 and aux_1. Here, aux_0 allows $\mathcal{A}$ to distinguish (perhaps by providing a differing input), but aux_1 provides no information that would allow $\mathcal{A}$ to distinguish, and in particular provides no auxiliary input. If aux were a uniform superposition of $\mathsf{aux}_0, \mathsf{aux}_1$, $\mathcal{A}$ can distinguish with inverse polynomial advantage, but the best success probability we could hope for from $\mathcal{E}$ is $1/2$, since it can only succeed for the portion of the superposition that is aux_0.

Thus, defining a Boyle et al.-style definition in the quantum setting is tricky. The challenges above are analogous to the challenges faced in defining traitor tracing quantumly, as observed by Zhandry [Zha20]. By adapting Zhandry's definition as given in Definition 4.7, we will obtain a strong and meaningful definition of diO for the quantum setting.

Our Definition. We are now ready to give our definition. Let $m(\lambda), n(\lambda), s(\lambda)$ be polynomials, and consider a quantum polynomial time algorithm $\mathsf{Samp}(1^\lambda)$ which samples (1) a pair of classical circuits C_0, C_1 of size $s(\lambda)$, input length $n(\lambda)$ and output length $m(\lambda)$, and (2) a *quantum* side information aux. Consider a distinguishing adversary $\mathcal{A}$. Consider the following operation on aux:

- Choose a random bit $b \leftarrow \{0,1\}$.
- Run $\hat{C} \leftarrow \mathsf{diO}(1^\lambda, C_b)$ to get an obfuscation of a random choice of C_0, C_1.
- Run $b' \leftarrow \mathcal{A}(\hat{C}, C_0, C_1, \mathsf{aux})$.
- Output 1 if and only if $b' = b$; otherwise output 0.

Let $\mathcal{M}^{\mathsf{diO}} = (M_0, M_1)$ be the POVM given by this operation, which we call the *associated POVM* to the sampler. $\mathcal{M}^{\mathsf{diO}}$ has a projective implementation $\mathsf{ProjImp}(\mathcal{M}^{\mathsf{diO}}) = \{M'_p\}_p$, where each M'_p corresponds to the probability distribution on $\{0,1\}$ that is 1 with probability p.

[4] [BCP14] also give a worst-case definition, and indeed show iO implies the worst-case notion. However, for this to hold, they need iO secure against *non-uniform* attackers. If we want to consider uniform security, we must consider average-case diO.

Now consider an extractor $\mathcal{E}$ which takes as input C_0, C_1, aux as well as a parameter ϵ whose reciprocal is given in unary (so $1^{1/\epsilon}$), so that if $\mathcal{E}$ runs in polynomial time, then it runs in time polynomial in λ whenever ϵ is an inverse polynomial in λ. We define the following experiment, parameterized by $\mathcal{S}, \mathcal{A}, \mathcal{E}$ as well as a function ϵ:

- Run $(C_0, C_1, \mathsf{aux}) \leftarrow \mathcal{S}(1^\lambda)$. Let $\mathcal{H}$ be the register containing aux.
- Apply $\mathsf{ProjImp}(\mathcal{M}^{\mathsf{diO}})$ to $\mathcal{H}$, resulting in probability p.
- If $p > \frac{1}{2} + \epsilon(\lambda)$, we say that event $\mathsf{Live}_\epsilon^{\mathsf{diO}}$ happens.
- Next, run $\mathcal{E}$ on $\mathcal{H}$, obtaining an input x or symbol $\perp$
- If $x \neq \perp$ but $C_0(x) = C_1(x)$, we say event $\mathsf{Fail}_\epsilon^{\mathsf{diO}}$ happens.
- If $x \neq \perp$ and $C_0(x) \neq C_1(x)$, we say event $\mathsf{Success}_\epsilon^{\mathsf{diO}}$ happens.

Definition 4.8 (Post-Quantum Differing-inputs Obfuscation)**.** A post-quantum *differing-inputs obfuscator* (diO) is a PPT algorithm diO such that:

- $\mathsf{diO}(1^\lambda, C)$ is equivalent to C with overwhelming probability over the randomness of diO.
- For every QPT adversary $\mathcal{A}$ and sampler $\mathcal{S}$, there exists a QPT $\mathcal{E}$ such that for every inverse polynomial function $\epsilon = \epsilon(\lambda)$, there exists a negligible $\mathsf{negl} = \mathsf{negl}(\lambda)$ such that $\Pr[\mathsf{Fail}_\epsilon^{\mathsf{diO}}] < \mathsf{negl}$ and $\Pr[\mathsf{Live}^{\mathsf{diO}} \wedge \neg\mathsf{Success}_\epsilon^{\mathsf{diO}}] < \mathsf{negl}$.

5 Hidden Partitions and the BOS Model

A partition Π of a set T is a collection of subsets $\Pi = \{S_1, \ldots, S_n\}$ such that $T = \cup_i S_i$ and the S_i are disjoint. The sets S_i are called parts of Π. Given partition $\Pi = \{S_1, \ldots, S_n\}$ of T and $\Pi' = \{S'_1, \ldots, S'_{n'}\}$ of T', we can define the product partition $\Pi \times \Pi'$ of $T \times T'$ as $\Pi \times \Pi' = \{S_i \times S'_j\}_{(i,j)\in[n]\times[n']}$. We can similarly define the product of several partitions.

The basic setup. Fix a set $\mathcal{Q}$, $\alpha \in \mathcal{Q}$ a distinguished element, and $\Omega \subseteq \mathcal{Q}$ a distinguished subset. Also fix a relation $R(\Pi, w)$ that takes as input partitions Π of $\mathcal{Q}$ and strings $w \in \{0,1\}^*$ and outputs a bit.

5.1 The Quantum Hidden Partition Problem

Given $\mathcal{Q}, \alpha, \Omega, R$ as above, consider a QPT $\mathcal{S}(1^\lambda)$, called a quantum hidden partition sampler (QHPS) that samples $(|P\rangle, \Pi, \mathcal{D}, \mathsf{aux}) \leftarrow \mathcal{S}(1^\lambda)$ such that:

- $|P\rangle$ is a quantum state program taking inputs in some domain $\mathcal{X}$ and outputting a bit.
- Π is a partition of $\mathcal{Q}$, and
- $\mathcal{D} : \mathcal{Q} \to \mathcal{X} \times \{0,1\}$ is a PPT algorithm mapping $\mathcal{Q}$ to $\mathcal{X} \times \{0,1\}$.

Now consider two experiments involving a QHPS $\mathcal{S}$ and a quantum adversary $\mathcal{A}$. The first, called the *sink indistinguishability* experiment, works as follows:

- First run $(|P\rangle, \Pi, \mathcal{D}) \leftarrow \mathcal{S}(1^\lambda)$, and give $|P\rangle$ to $\mathcal{A}$.
- Throughout, $\mathcal{A}$ may make *classical* queries to $\mathcal{D}$, sending $q \in \mathcal{Q}$, and receiving independent samples $(x, b) \leftarrow \mathcal{D}(q)$. Repeated queries on the same q will give independent samples.
- Eventually $\mathcal{A}$ outputs a sink $q^* \in \Omega$. In response, run $(x^*, b^*) \leftarrow \mathcal{D}(q^*)$ and send x^* to $\mathcal{A}$.
- $\mathcal{A}$ outputs a guess b' for b^*. The experiment outputs $o = b' \oplus b^*$;

We say $\mathcal{A}$ wins the sink indistinguishability experiment if $o = 0$ (equivalently, $b' = b^*$).

The second experiment, called the *partition indisitnguishability* experiment, works as follows:

- First run $(|P\rangle, \Pi, \mathcal{D}) \leftarrow \mathcal{S}(1^\lambda)$, and give $|P\rangle$ to $\mathcal{A}$.
- Throughout, $\mathcal{A}$ may make *classical* queries to $\mathcal{D}$, sending $q \in \mathcal{Q}$, and receiving independent samples $(x, b) \leftarrow \mathcal{D}(q)$.
- Eventually $\mathcal{A}$ outputs two queries q_0^*, q_1^*. If q_0^*, q_1^* are in the same part of Π, output a random bit o and abort. Otherwise, choose a random bit c and run $(x, b) \leftarrow \mathcal{D}(q_c^*)$ and send (x^*, b^*) to $\mathcal{A}$.
- $\mathcal{A}$ outputs a guess c' for c. The experiment outputs $o = c' \oplus c$; $\mathcal{A}$ wins if $o = 0$.

We say $\mathcal{A}$ wins the partition indistinguishability experiment if $o = 0$.

Definition 5.1. $\mathcal{S}$ is a *valid* QHPS if for any QPT adversaries $\mathcal{A}_0, \mathcal{A}_1$, there exists negligible functions $\mathsf{negl}_0, \mathsf{negl}_1$ such that the probability $\mathcal{A}_0$ wins the sink indistinguishability experiment is at most $\mathsf{negl}_0(\lambda)$, and the probability $\mathcal{A}_1$ wins the partition indistinguishability experiment is at most $\mathsf{negl}_1(\lambda)$.

Consider the POVM which runs $(x, b) \leftarrow \mathcal{D}(\alpha)$, and run $|P\rangle(x)$ to obtain a bit b'. The POVM outputs 1 if $b' = b$. This POVM has a projective implementation, which we denote $\mathsf{ProjImp}(\mathcal{D}(\alpha))$. Now, consider the following experiment with algorithm $\mathcal{T}$:

- First run $(|P\rangle, \Pi, \mathcal{D}) \leftarrow \mathcal{S}(1^\lambda)$.
- Next apply $\mathsf{ProjImp}(\mathcal{D}(\alpha))$ to the register $\mathcal{H}$ containing $|P\rangle$, obtaining probability p.
- Run $\mathcal{T}$ on $1^{1/\epsilon}$ and $\mathcal{H}$. $\mathcal{T}$ can make queries to $\mathcal{D}$. Importantly, on query q, $\mathcal{T}$ can obtain a superposition of samples from $\mathcal{D}(q)$. The output is a string w or an abort symbol $\perp$.

Let $\mathsf{Live}_\epsilon^{\mathsf{BOS}}$ be the event that $p \geq \frac{1}{2} + \epsilon$. The event $\mathsf{Live}_\epsilon^{\mathsf{BOS}}$ corresponds to $|P\rangle$ being able to predict the bit b with advantage at least ϵ. Let $\mathsf{Success}_\epsilon^{\mathsf{BOS}}$ be the event that $\mathcal{T}$ outputs a $w \neq \perp$ such that $R(\Pi, w) = 1$, and let $\mathsf{Fail}_\epsilon^{\mathsf{BOS}}$ be the event that $\mathcal{T}$ outputs a $w \neq \perp$ such that $R(\Pi, w) = 0$.

Definition 5.2. $\mathcal{Q}, \alpha, \Omega, R$ is *solvable* if there exists a $\mathcal{T}^{\mathcal{D}}(|P\rangle, \epsilon)$ that runs in time polynomial in λ and $1/\epsilon$ and makes quantum queries to $\mathcal{D}$, such that for any valid QHPS $\mathcal{S}$ and inverse-polynomial ϵ there is a negligible negl such that $\Pr[\mathsf{Live}_\epsilon^{\mathsf{BOS}} \wedge \neg \mathsf{Success}_\epsilon^{\mathsf{BOS}}] \leq \mathsf{negl}(\lambda)$ and $\Pr[\mathsf{Fail}_\epsilon^{\mathsf{BOS}}] \leq \mathsf{negl}(\lambda)$.

In other words, $\mathcal{T}$ should almost always succeed if $|P\rangle$ starts out live, and $\mathcal{T}$ should almost never output a w that is not accepted by R (outputting $\perp$ instead if it cannot succeed).

5.2 The BOS Model

Now consider a stateful interactive potentially randomized algorithm O which takes as a secret input a partition Π, and then receives a sequence of queries $q_1, q_2, \cdots \in \mathcal{Q}$ and produces a corresponding sequence of outputs $o_1, o_2, \cdots \in \{0,1\}$. We will denote an interactive algorithm $\mathcal{A}$ interacting with O and outputting w as $w \leftarrow \mathcal{A} \Leftrightarrow O(\Pi)$.

Definition 5.3. *O is a BOS oracle* if, for any partition Π and any poly-length sequence of queries $q_1, q_2, \ldots$, $O(\Pi)$ satisfies each of the following guarantees:

- **Accepts first distinguished query.** If $q_1 = \alpha$, then $o_1 = 1$.
- **Rejects sinks.** For any i, if $q_i \in \Omega$, then $o_i = 0$.
- **Local consistency.** For any two consecutive queries q_i, q_{i+1}, if $q_i, q_{i+1} \in P \in \Pi$ (that is, if q_i, q_{i+1} are in the same part of the partition Π), then $o_i = o_{i+1}$.
- **Single step rewinding.** For any i, if $q_{i+2} = q_i$, then $o_{i+2} = o_i$.

Definition 5.4. A PPT algorithm $\mathcal{A}$ *solves* R (with the associated $\mathcal{Q}, \alpha, \Omega$) in the BOS model if, for any BOS oracle O and any partition Π of Q, then $\Pr[R(\Pi, w) = 1 : w \leftarrow A \Leftrightarrow O(\Pi)] = 1$.

5.3 From BOS to Solving Quantum Hidden Partitions

Here, we present the main technical tool of this paper:

Theorem 5.5. *For any $R, \mathcal{Q}, \alpha, \Omega$, if there exists an algorithm $\mathcal{A}$ which solves R in the BOS model in polynomial time, then $R, \mathcal{Q}, \alpha, \Omega$ is solvable.*

We now prove Theorem 5.5. We first assume without loss of generality that $\mathcal{A}$ has the following properties:

- $q_1 = \alpha$. To see why this is without loss of generality, let $\mathcal{A}$ solve R in the BOS model, and let $\mathcal{A}'$ be $\mathcal{A}$, except that if the first query is *not* α, $\mathcal{A}'$ inserts a dummy query to α as the first query, and then ignores the response. It is easy to see that the oracle seen by $\mathcal{A}$ as a subroutine of $\mathcal{A}'$ is still a BOS algorithm, and so $\mathcal{A}'$ still solves R.
- If $o_i = 0$, then $q_{i+1} = q_{i-1}$. A simple inductive argument then shows that any $o_i = 0$ is always preceded by an $o_{i-1} = 1$. To see why this is without loss of generality, let $\mathcal{A}$ solve R in the BOS model, and let $\mathcal{A}'$ be $\mathcal{A}$ except that if $o_i = 0$ and $q_{i+1} \neq q_{i-1}$, then $\mathcal{A}'$ makes a final query on q_{i-1}, immediately stops making queries (including not querying on q_{i+1}), and answers every subsequent query by $\mathcal{A}$ with 0. The oracle seen by $\mathcal{A}$ as a subroutine of $\mathcal{A}'$ is still a BOS algorithm, so $\mathcal{A}'$ still solves R. But the new $\mathcal{A}'$ will always query on $q_{i+1} = q_{i-1}$ if $o_i = 0$.

We call an $\mathcal{A}$ with the above properties *normal form*. We describe the algorithm $\mathcal{T}$ given a normal form $\mathcal{A}$:

Algorithm 5.6. Let $\mathcal{A}$ be any normal form algorithm in the BOS model. Consider a quantum program $|P\rangle$ stored in register $\mathcal{H}$. First, define the following:

- Let r be an upper bound on the number of queries made by $\mathcal{A}$.

- Let $\epsilon' = \epsilon/2r, \delta = 2^{-\lambda}, T = 1/\sqrt{\delta}$.
- Let $\mathsf{Pred} = \{\mathsf{Pred}_x\}_x$ be the collection of projective measurements applied to $\mathcal{H}$ corresponding to running $|P\rangle$ on input x.
- Let $\mathsf{EST}(q)$ be the algorithm $\mathsf{API}^{\mathsf{Pred},\mathcal{D}(q)}_{\epsilon'/4,\delta}$, where API is the algorithm in Lemma 3.3. We will dilate $\mathsf{EST}(q)$ so that it is a projective measurement. This means that $\mathsf{EST}(q)$ acts on register $\mathcal{H} \times \mathcal{I}_q$, where $\mathcal{I}_q$ are the anilla registers used to purify, with a different register used for every query.

We now give the algorithm $\mathcal{T}^{\mathcal{D}}(|P\rangle, 1^{1/\epsilon})$. Run $\mathcal{A}$, which makes queries $q_1, \ldots, q_r$. To answer each query q_i, do the following:

1. Define $p_{q_0} = 1/2 + \epsilon$.
2. If $q_i \neq q_{i-2}$:
   (a) Create the register $\mathcal{I}_{q_i}$, replacing any existing register with that name.
   (b) Then run $\mathsf{EST}(q_i)$, obtaining measurement outcome p_{q_i}, replacing any existing value for p_{q_i}.
   (c) If $p_{q_i} \leq p_{q_{i-1}} - \epsilon'$, answer the query with $o_{q_i} = 0$.
   (d) If $p_{q_i} > p_{q_{i-1}} - \epsilon'$, answer the query with $o_{q_i} = 1$.
   (e) If $i = 1$ and $o_{q_1} = 0$ (that is, this is the first query and the query response is 0) then immediately abort and output $\perp$.
3. Otherwise, if $q_i = q_{i-2}$:
   (a) Run the algorithm $\mathsf{Repair}^{\mathsf{EST}(q_{i-1}),\mathsf{EST}(q_{i-2})}_{T,p_{q_{i-1}},p_{q_{i-2}}}$.
   (b) Update p_{q_i} to $p_{q_{i-2}} - \epsilon'$, and answer the query with $o_{q_i} = o_{q_{i-2}}$.

When $\mathcal{A}$ terminates and outputs w, $\mathcal{T}$ outputs w.

We now prove that $\mathcal{T}$ solves $\mathcal{Q}, \alpha, \Omega, R$. We first need the following lemma:

Lemma 5.7. *Suppose $\mathcal{T}$ does not abort in Step 2e. Then except with negligible probability, every query q_i that $\mathcal{T}$ responds with 1 will have $p_{q_i} \geq 1/2 + \epsilon - i \times \epsilon'$.*

Proof. We prove by induction on i. Since $p_{q_0} = 1/2 + \epsilon$, $\mathcal{T}$ does not abort in Step 2e only if $p_{q_1} > p_{q_0} - \epsilon' = 1/2 + \epsilon - 1 \times \epsilon'$. Now we inductively assume the lemma holds for all queries before query i, and prove it holds for query i. We break into three cases:

- $q_i = q_{i-2}$. In this case, $o_{q_i} = o_{q_{i-2}}$, so if $\mathcal{T}$ responds with 1, then $\mathcal{T}$ must have responded two queries ago with 1 as well. By the inductive hypothesis, this means $p_{q_{i-2}} \geq 1/2 + \epsilon - (i-2) \times \epsilon'$. But now we update p_{q_i} to $p_{q_{i-2}} - \epsilon' \geq 1/2 + \epsilon - (i-1) \times \epsilon' \geq 1/2 + \epsilon - i \times \epsilon'$.
- $q_i \neq q_{i-1}$, and $o_{q_{i-1}} = 1$. In this case, by induction, we have $p_{q_{i-1}} \geq 1/2 + \epsilon - (i-1) \times \epsilon'$. If $o_{q_i} = 1$, then it must be that $q_i \geq q_{i-1} - \epsilon' \geq 1/2 + \epsilon - i \times \epsilon'$.
- $q_i \neq q_{i-2}$ and $o_{q_{i-1}} = 0$. By the assumption that $\mathcal{A}$ is valid, this is impossible.

This completes the proof of Lemma 5.7. □

We now prove the following lemma:

Lemma 5.8. *If $\mathcal{T}$ does not abort in Step 2e, then except with negligible probability the oracle $\mathcal{T}$ presents to $\mathcal{A}$ is a BOS oracle.*

Proof. We prove the properties of a BOS oracle.

- **First distinguished query**: Recall we assumed $q_1 = \alpha$. If $\mathcal{T}$ does not abort in Step 2e, then $\mathcal{T}$ responds to the query with 1.
- **Sinks**: Consider a query $q_i \in \Omega$. We claim that $\mathsf{EST}(q_i)$ gives measurement outcome p_{q_i} that is at most $\frac{1}{2} + \epsilon'$, except with negligible probability. First consider replacing $\mathsf{EST}(q_i) = \mathsf{API}^{\mathsf{Pred},\mathcal{D}(q_i)}_{\epsilon'/4,\delta}$ (which produces p_{q_i}) with $\mathsf{ProjImp}(\mathcal{D}(q_i))$, giving outcome p'. By Lemma 3.3, since $\Delta_{\epsilon'/4}(\mathsf{EST}(q_i), \mathsf{ProjImp}(\mathcal{D}(q_i)) \leq \delta$, we have $|p' - p_{q_i}| < \epsilon'/4$ except with negligible probability.

  Then we note that p' must be at most $1/2 + \mathsf{negl} \leq 1/2 + \epsilon'/4$, except with negligible probability. Indeed, if this were not the case, then the bit b produced by $\mathcal{D}(q)$ could be predicted with non-negligible probability. Thus we have that $|p_{q_i} - \frac{1}{2}| \leq \epsilon'/2 \leq \epsilon'$. Now that $p_{q_i} \leq \frac{1}{2} + \epsilon'$, we use Lemma 5.7. If $o_{q_i} = 1$, then $p_{q_i} \geq 1/2 + \epsilon - i\epsilon' \geq 1/2 + \epsilon - r\epsilon' > 1/2 + \epsilon'$ since $(r+1)\epsilon' < \epsilon$, we therefore have that o_{q_i} must be 0.
- **Local consistency**: Suppose q_i, q_{i+1} lie in the same part of Π. We claim that the measurement outcomes p_{q_i} and $p_{q_{i+1}}$ satisfy $p_{q_{i+1}} \geq p_{q_i} - \epsilon'$. Indeed, consider replacing $\mathsf{EST}(q_{i+1}) = \mathsf{API}^{\mathsf{Pred},\mathcal{D}(q_{i+1})}_{\epsilon'/4,\delta}$ (which produces $p_{q_{i+1}}$) with each of the following:
  - $\mathsf{EST}(q_i) = \mathsf{API}^{\mathsf{Pred},\mathcal{D}(q_i)}_{\epsilon'/4,\delta}$, giving outcome p'. Since the last measurement on $\mathcal{H}$ was also $\mathsf{EST}(q_i)$, by being almost projective via Lemma 3.3, we have $|p' - p_{q_i}| < \epsilon'/4$ except with negligible probability.
  - $\mathsf{ProjImp}(\mathcal{D}(q_i))$, giving outcome p''. Again by Lemma 3.3, since $\Delta_{\epsilon'/4}(\mathsf{EST}(q_i), \mathsf{ProjImp}(\mathcal{D}(q_i)) \leq \delta$, we have $|p'' - p'| < \epsilon'/4$ except with negligible probability.
  - $\mathsf{ProjImp}(\mathcal{D}(q_{i+1}))$, giving p'''. Since q_i, q_{i+1} are in the same part of Π, the distributions $\mathcal{D}(q_i)$ and $\mathcal{D}(q_{i+1})$ are computationally indistinguishable. Therefore, by Lemma 3.4, $|p''' - p'| \leq \epsilon'/4$ except with negligible probability.
  - $\mathsf{EST}(q_{i+1})$, giving $p_{q_{i+1}}$. As above, by Lemma 3.3, we have $|p_{q_{i+1}} - p'''| \leq \epsilon'/4$ except with negligible probability. Putting the above inequalities together shows that $|p_{q_{i+1}} - p_{q_i}| \leq \epsilon'$ except with negligible probability.
- **Single-step Rewinding**: If $q_i = q_{i-2}$, then Lemma 3.2 and the fact that $\mathsf{EST}(q_i)$ is almost projective immediately gives us that $|p_{q_i} - p_{q_{i-2}}| \leq \epsilon'/2 \leq \epsilon'$ except with probability $N\delta + N/T + 4\sqrt{\delta} = N\delta + (N+4)\sqrt{\delta}$, which is negligible.

One issue with the above proof is that in order to actually turn a bit predictor in the sink proof or a distinguisher in the local consistency proof into algorithms for the sink and partition indistinguishability games, the adversary needs to be able to run API, which in turn needs to get

superpositions of samples from $\mathcal{D}(q)$. While $\mathcal{T}$ is allowed such queries, we do not allow adversaries for the sink and partition indistinguishability games such quantum access. However, following Zhandry [Zha12], we can simulate such superpositions of samples with a polynomial number of samples, thus getting an algorithm for these games which only makes classical queries. This completes the proof of Lemma 5.8. □

Now suppose $\mathsf{Live}_\epsilon^{\mathsf{BOS}}$ happens. In this case, by Lemma 3.3, $\Delta_{\epsilon'/4}(\mathsf{EST}(\alpha), \mathsf{ProjImp}(\mathcal{D}(\alpha))) \leq \delta$. This means, except with negligible probability δ, $p_\alpha = p_{q_1}$ will be at least $\epsilon - \epsilon'/4 \geq p_{q_0} - \epsilon'$. Thus, $\mathcal{T}$ will not abort in Step 2e. Therefore, by Lemma 5.8, $\mathcal{T}$ presents a BOS oracle to $\mathcal{A}$, meaning $\mathcal{A}$ outputs a w such that $R(\Pi, w) = 1$. Thus $\mathsf{Live}_\epsilon^{\mathsf{BOS}} \wedge \neg\mathsf{Success}_\epsilon^{\mathsf{BOS}}$ happens except with negligible probability.

We now turn to proving that $\mathsf{Fail}_\epsilon^{\mathsf{BOS}}$ happens with negligible probability. For $\mathsf{Fail}_\epsilon^{\mathsf{BOS}}$ to happen, we must have (1) that $\mathcal{T}$ does *not* abort in Step 2e, and (2) that $\mathcal{T}$ fails to output a w such that $R(\Pi, w) = 1$. But by Lemma 5.8, if $\mathcal{T}$ does not abort, then it presents a BOS oracle to $\mathcal{A}$ except with negligible probability, meaning $R(\Pi, w) = 1$. Thus $\mathsf{Fail}_\epsilon^{\mathsf{BOS}}$ happens with negligible probability. This completes the proof of Theorem 5.5. □

6 The Quantum iO to diO Transformation

In this section, we prove that iO implies diO for circuits with a polynomial number of differing inputs. This was shown classically by [BCP14], and we show that the same result holds quantumly as well using our BOS model.

6.1 The Hidden Partition

Here we give the *boundary* hidden partition, denoted with the superscript $^{\mathsf{Bnd}}$. Let $N = 2^n$ and we interpret the domain $\{0,1\}^n$ as the interval $[N]$. We set $\mathcal{Q}^{\mathsf{Bnd}} = [0, N], \alpha^{\mathsf{Bnd}} = N, \Omega^{\mathsf{Bnd}} = \{0\}$. We say a partition Π of $\mathcal{Q}^{\mathsf{Bnd}}$ is *contiguous* if each part of Π is an interval $[a, b]$, so that $\Pi = \{[0, a_1 - 1], [a_1, a_2 - 1], \ldots, [a_{k-1}, a_k - 1], [a_k, N]\}$. For a contiguous partition, we say that $a_1, \ldots, a_k$ are the boundaries. Let $R^{\mathsf{Bnd}}(\Pi, w)$ be the following relation:

- Output 1 if Π is *not* contiguous.
- Output 1 if Π is contiguous and w is a boundary of Π.
- Output 0 if Π is contiguous but w is not a boundary of Π.

The first condition means that we trivially win for all non-contiguous Π, and can focus on the case of contiguous Π, where the goal is to find a boundary of Π.

We now explain how diO leads to an instance of the quantum hidden partition problem relative to $\mathcal{Q}^{\mathsf{Bnd}}, \alpha^{\mathsf{Bnd}}, \Omega^{\mathsf{Bnd}}, R^{\mathsf{Bnd}}$. We interpret a circuit sampler $\mathcal{S}$ and adversary $\mathcal{A}$ as a quantum hidden partition sampler $\mathcal{S}^{\mathsf{diO}}$, where $|P\rangle(\hat{C})$ is the algorithm that runs $\mathcal{A}$ on aux and $\hat{C}$. The algorithm $\mathcal{D}$, on input $q \in \mathcal{Q}^{\mathsf{Bnd}}$, chooses a random bit b, and then outputs $(\hat{C}_b, b)$, where $\hat{C}_0$ is an obfuscation of C_0, and $\hat{C}_1$ is an obfuscation of the program

$$C^{(0)} = C_0$$
$$C^{(q)}(x) = \begin{cases} C_1(x) & \text{if } x \le q \\ C_0(x) & \text{if } x > q \end{cases} \quad \text{for } 0 < q < N$$
$$C^{(N)} = C_1$$

Finally, the partition Π is the contiguous partition whose boundaries are exactly the differing inputs of C_0, C_1.

Lemma 6.1. *Assuming* diO *is a secure* iO*, the QHPS* $\mathcal{S}^{\mathsf{diO}}$ *is valid with respect to* $\mathcal{Q}^{\mathsf{Bnd}}, \alpha^{\mathsf{Bnd}}, \Omega^{\mathsf{Bnd}}, R^{\mathsf{Bnd}}$.

Proof. The unique sink is 0, and note that $C^{(0)} = C_0$, and so the response $\hat{C}$ to a query on $q = 0$ is independent of the bit b. Thus the probability any adversary can predict b (and therefore win the sink indistinguishability experiment) is exactly $1/2$.

Next, observe that $C^{(q)}$ is equivalent to $C^{(q')}$ for $q' > q$ as long as C_0, C_1 have no differing inputs in the interval $[q, q' - 1]$. Thus, if there are no differing inputs in $[q, q' - 1]$, by iO we have that no efficient adversary can distinguish query responses from the same part of Π. Therefore, the probability of winning the partition indistinguishability experiment is $1/2 + \mathsf{negl}$. □

6.2 The BOS Algorithm

Algorithm 6.2 (BOS Algorithm $\mathcal{A}^{\mathsf{Bnd}}$)**.** Initialize integers $a, b \in [0, N]$. Set $a = N, b = 0$, and query on a, which by definition gives response $o = 1$. Then do the following for at most $O(k \log^2 N)$ steps, where k is an upper bound on the number of parts in Π:

1. Query on b, obtaining response o.
2. If $o = 1$: set $(a, b) = (b, 0)$ and go to Step 1.
3. Else $(o = 0)$:
   (a) If $b \neq a - 1$: Query on a, set $b = \lfloor (a + b)/2 \rfloor$ (a remains unchanged), and go to Step 1.
   (b) Else $(b = a - 1)$: output a.

If the algorithm has not terminated in $O(k \log^2 N)$ steps, output $\perp$.

Theorem 6.3. $\mathcal{A}^{\mathsf{Bnd}}$ *solves* $(Q^{\mathsf{Bnd}}, \alpha^{\mathsf{Bnd}}, \Omega^{\mathsf{Bnd}}, R^{\mathsf{Bnd}})$ *in the BOS model.*

Proof. We first observe the following invariants:

- Any time the algorithm goes to Step 1, the most recent previous query was on a. This is because either we return to Step 1 following Step 3a which queries a, or because we return to Step 1 in Step 2, which sets a to b, after having just queried b.
- At *any* point, one of the most recent two queries was on a.
- The only time a is updated is in Step 2, immediately after which the most recent query on a yielded 1.

- By local consistency and single step rewinding, at all times the most recent query on a yielded 1.
- Whenever the algorithm terminates in Step 3b, the most recent queries were on a and $b = a-1$, and the responses were 1 and 0 respectively. By local consistency, this means a is a boundary.

To prove correctness, it therefore suffices to guarantee that A_{Boundary} eventually terminates in Step 3b, which is equivalent to showing that it reaches Step 3b in at most $O(k \log^2 N)$ steps given a contiguous partition.

Toward that end, we observe that every time Step 3 is invoked, either we terminate in Step 3b or the difference $a - b$ is approximately halved. Thus Step 3 can only be called in at most $O(\log N)$ consecutive iterations without terminating in Step 3b. Moreover, by local consistency, if we invoke Step 3, it must be that there is a boundary in the interval $[b, a]$, and so after completing Step 3a (which updates b), there must be a boundary in $[b - (a - b), a]$.

On the other hand, after every call to Step 2 must follow a call to Step 3, since b was set to 0, which is guaranteed to have query response 0. Moreover, every time we call Step 2, we there must have been a boundary in the interval $[b - (a - b), a]$ (before updating a, b), because the previous iteration would have called Step 3. Thus, if we let c be the closest boundary less than a, every call to Step 2 must either (1) at least halve the distance between a and c (2) set a to be less than c. (1) can only happen $O(\log N)$ times consecutively, and (2) can only happen k times in total since there are only k boundaries. Thus the total number of times Step 2 is called is at most $O(k \log N)$.

Putting everything together gives an upper bound of $O(k \log^2 N)$ total steps before terminating in Step 3b, thus proving the correctness of Algorithm A^{Bnd}. □

Corollary 6.4. *Any iO is also diO for $\mathcal{S}$ where the number of differing inputs is bounded by a polynomial.*

Proof. We note that $\mathsf{Live}^{\mathsf{diO}}, \mathsf{Success}^{\mathsf{diO}}, \mathsf{Fail}^{\mathsf{diO}}$ are identical to the events $\mathsf{Live}^{\mathsf{BOS}}$, $\mathsf{Success}^{\mathsf{BOS}}$ and $\mathsf{Fail}^{\mathsf{BOS}}$. Then combining Algorithm $\mathcal{A}^{\mathsf{Bnd}}$ with Theorem 5.5 gives extractor $\mathcal{E} = \mathcal{T}$, proving diO. □

7 Post-Quantum Tracing with Embedded Identities

Here, we show how to build embedded identity traitor tracing from functional encryption, specifically a special case called private linear broadcast encryption (PLBE). PLBE with polynomial index space was first formalized by [BSW06] to build ordinary traitor tracing in the classical setting, and the result was upgraded to the quantum setting by [Zha20], though only in the setting of polynomial identity spaces. Using PLBE with exponential index space to build embedded identity traitor tracing was proposed by [NWZ16] for the classical setting. Here, we upgrade their result to the quantum setting.

Construction 7.1. Let Π^{FE} be a functional encryption scheme. We construct the traitor tracing scheme $\Pi^{\mathsf{TT}} = (\mathsf{Gen}^{\mathsf{TT}}, \mathsf{Derive}^{\mathsf{TT}}, \mathsf{Enc}^{\mathsf{TT}}, \mathsf{Dec}^{\mathsf{TT}}, \mathsf{Trace}^{\mathsf{TT}})$, where we define $\mathsf{Gen}^{\mathsf{TT}}, \mathsf{Derive}^{\mathsf{TT}}, \mathsf{Enc}^{\mathsf{TT}}, \mathsf{Dec}^{\mathsf{TT}}$ below:

- $\mathsf{Gen}^{\mathsf{TT}}(1^\lambda, N) = \mathsf{Gen}^{\mathsf{FE}}(1^\lambda)$
- $\mathsf{Derive}^{\mathsf{TT}}(\mathsf{msk}, \mathsf{id}) = \mathsf{Derive}^{\mathsf{FE}}(\mathsf{msk}, f_{\mathsf{id}})$ where $f_{\mathsf{id}}(x, m) = \begin{cases} m & \text{if } x \geq \mathsf{id} \\ \bot & \text{otherwise} \end{cases}$

- $\mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m) = \mathsf{Enc}^{\mathsf{FE}}(\mathsf{pk}, (N, m))$
- $\mathsf{Dec}^{\mathsf{TT}}(\mathsf{sk}_{\mathsf{id}}, c) = \mathsf{Dec}^{\mathsf{FE}}(\mathsf{sk}_{\mathsf{id}}, c)$

7.1 The Hidden Partition

We use the same hidden partition $\mathcal{Q}^{\mathsf{Bnd}}, \alpha^{\mathsf{Bnd}}, \Omega^{\mathsf{Bnd}}, R^{\mathsf{Bnd}}$ as in Section 6. We now explain how Construction 7.1 leads to an instance of the quantum hidden partition problem relative to this partition. We interpret an adversary $\mathcal{A}^{\mathsf{TT}}$ interacting in the tracing experiment as a quantum hidden partition sampler $\mathcal{S}^{\mathsf{TT}}$, where $|P\rangle$ is the decoder $|\mathsf{D}\rangle$ outputted by $\mathcal{A}^{\mathsf{TT}}$, Π is the contiguous partition whose boundaries are the identities queried by $\mathcal{A}^{\mathsf{TT}}$. Finally $\mathcal{D}$, on input q, encrypts (q, m_b) for a random choice of b to get ciphertext c, and outputs (c, b).

Lemma 7.2. *Assuming Π^{FE} is an adaptively secure FE scheme, the QHPS $\mathcal{S}^{\mathsf{TT}}$ is valid for $\mathcal{Q}^{\mathsf{Bnd}}, \alpha^{\mathsf{Bnd}}, \Omega^{\mathsf{Bnd}}, R^{\mathsf{Bnd}}$.*

Proof. The unique sink is 0, and note that encryptions of $(0, m)$ computationally hide m, since the only secret keys are f_{id} for $\mathsf{id} > 0$. Thus the probability the adversary can predict b (and therefore win the sink indistinguishability experiment) is at most $1/2 + \mathsf{negl}$.

Next, observe that encryptions of (q, m) and (q', m) for $q' > q$ decrypt identically under any secret key except those for id in the interval $[q, q' - 1]$. Thus, by FE security, the encryptions are indistinguishable unless the adversary has an $\mathsf{id} \in [q, q' - 1]$, meaning no efficient adversary can distinguish query responses from the same part of Π. Therefore, the probability of winning the partition indistinguishability experiment is $1/2 + \mathsf{negl}$. □

Corollary 7.3. *Assuming Π^{FE} is an adaptively secure FE scheme, Construction 7.1 is quantum traceable.*

Proof. Let $\mathcal{T}(|P\rangle, 1^{1/\epsilon})$ be the algorithm guaranteed by Theorem 5.5 and the existence of $\mathcal{A}^{\mathsf{Bnd}}$. Then set $\mathsf{Trace}^{\mathsf{TT}}(\mathsf{pk}, m_0, m_1, 1^{1/\epsilon}, |\mathsf{D}\rangle)$ to be $\mathcal{T}^{\mathcal{D}}(|\mathsf{D}\rangle, 1^{1/\epsilon})$ where $\mathcal{D}$ is the sampler above obtained from pk, m_0, m_1. Now consider the events $\mathsf{Live}_\epsilon^{\mathsf{BOS}}, \mathsf{Success}_\epsilon^{\mathsf{BOS}}, \mathsf{Fail}_\epsilon^{\mathsf{BOS}}$ for the QHPS described above, and the events $\mathsf{Live}_\epsilon^{\mathsf{TT}}, \mathsf{Success}_\epsilon^{\mathsf{TT}}, \mathsf{Fail}_\epsilon^{\mathsf{TT}}$ for the tracing experiment with Π^{TT} from Construction 7.1. We see that the events exactly coincide; in particular $D(\alpha)$ is identical to running $\mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m_b)$ for a random choice of b. Thus, by the guarantees of Theorem 5.5, Π^{TT} in Construction 7.1 is secure. □

8 Traitor Tracing from Collusion-Resistant IPP Codes

In [BN08, Sir06, BP08], it was shown how to construct collusion-secure traitor tracing with constant-sized ciphertexts from binary fingerprinting codes [BS95]. The scheme, described momentarily, naturally generalizes to larger alphabets (at the cost of larger ciphertexts). However, the code needed for the generalization is not a fingerprinting code, but a collusion resistant identifiable parent property (IPP) code [HvLT98], which has a different marking condition. It turns out that IPP codes and fingerprinting codes coincide for binary codes. But by generalizing to larger alphabets (and using IPP codes) we can abstract other existing combinatorial schemes in the literature such as [CFN94].

We now recall the scheme, which is parameterized by a parameter t.

Construction 8.1. Let $(\mathsf{Gen}^{\mathsf{PK}}, \mathsf{Enc}^{\mathsf{PK}}, \mathsf{Dec}^{\mathsf{PK}})$ be a public key encryption scheme and $(\mathsf{Gen}^{\mathsf{IPP}}, \mathsf{Trace}^{\mathsf{IPP}})$ a δ-robust fingerprinting code. Define the following algorithms, which depend on a parameter σ that may be a function of δ, λ, ℓ:

- $\mathsf{Gen}^{\mathsf{TT}}(1^\lambda, 1^N, 1^c)$ [5]: Run $(\mathsf{tk}', w_1, \ldots, w_N) \leftarrow \mathsf{Gen}^{\mathsf{IPP}}(1^\lambda, 1^N, 1^c)$. For $j \in [\ell]$ and $\sigma \in \Sigma$, run $(\mathsf{ek}_{j,\sigma}, \mathsf{dk}_{j,\sigma}) \leftarrow \mathsf{Gen}^{\mathsf{PK}}(\lambda)$. Output

$$\begin{aligned}\mathsf{pk} &= \{\mathsf{ek}_{j,\sigma}\}_{j\in[\ell],\sigma\in\Sigma} \text{ (the public key)}\\ \mathsf{tk} &= (\{\mathsf{dk}_{j,\sigma}\}_{j\in[\ell],\sigma\in\Sigma}, \mathsf{tk}') \text{ (the tracing key)}\\ \mathsf{sk}_i &= (\{\mathsf{dk}_{j,w_{i,j}}\}_{j\in[\ell]}, w_i) \text{ for } i \in [N] \text{ (the secret key for user } i)\end{aligned}$$

- $\mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m)$: Choose a random subset $T \subseteq [\ell]$ of size t. Let $m_j, j \in T$ be a t-out-of-t secret sharing of m: m_j are uniform conditioned on $\oplus_{j\in T} m_j = m$. For each $j \in T, \sigma \in \Sigma$, compute $c_{j,\sigma} = \mathsf{Enc}^{\mathsf{PK}}(\mathsf{ek}_{j,\sigma}, m_j)$. Output $c = (T, \{c_{j,\sigma}\}_{j\in T,\sigma\in\Sigma})$.
- $\mathsf{Dec}^{\mathsf{TT}}(\mathsf{sk}_i, c)$: For each $j \in T$, run $m'_j \leftarrow \mathsf{Dec}^{\mathsf{PK}}(\mathsf{dk}_{j,w_{i,j}}, c_{j,w_{i,j}})$. Then output $m' = \oplus_{j\in T} m'_j$.

Quantum Tracing Challenges. In the classical setting, tracing works by first extracting a codeword w^* from the decoder D, and then traces w^* using the fingerprinting code. The first step is accomplished roughly by replacing $c_{i,\sigma}$ with junk and seeing if D still decrypts. Based on this information, the tracer can determine if w^*_i should be one of the symbols in Σ or ?. By doing this for each $i \in [\ell]$, the tracer extracts an entire codeword w^*.

We can easily mimic the above strategy in the quantum setting to derive a measurement for each i which computes w^*_i. The problem is that the measurements for each i might be incompatible, which means that the tracing algorithm cannot apply these measurements simultaneously. If applied sequentially, it could be that after computing the first several positions, the decoder becomes dead. There is no guarantee that an entire codeword w^* can be computed.

More abstractly, any tracing algorithm that works within the globally consistent hidden partition model is likely only able to extract logarithmically many bits, which is insufficient for obtaining a full codeword for the fingerprinting code.

8.1 The Hidden Partition

We will assume the alphabet Σ is equal to $[1, s]$. Let $\mathcal{Q}^{\mathsf{Prod}} = [0, s]^\ell$ and $\alpha^{\mathsf{Prod}} = s^\ell$. Let $\Omega^{\mathsf{Prod}}_\delta$ be the set of vectors w where the number of 0's is more than $\delta\ell$. We call a partition Π of $\mathcal{Q}$ valid if it is equal to a product partition $\Pi_1 \times \cdots \times \Pi_\ell$ where each Π_i is a contiguous partition of $[0, s]$ (recall contiguous partitions defined in Section 6).

Given a valid partition, let $(a_{i,j})_j$ for $i \in [\ell]$ be the boundaries of Π_i. Let $R^{\mathsf{Prod}}_\delta(\Pi, w)$ be the following relation:

- Output 1 if Π is *not* valid.
- Output 1 if (1) Π is valid with boundaries $(a_{i,j})_{i,j}$, (2) for all $i \in [\ell]$, $w_i =?$ or $w_i \in (a_{i,j})_j$, and (3) the number of ? in w is at most $\delta\ell$.
- Output 0 in all other cases.

[5] Recall that N is the total identity space, and c is the collusion bound.

The first condition means that we trivially win for all non-valid Π, and can focus on the case of valid Π, where the goal is to find a string w that does not have too many ? and matches the boundaries of the product partition.

We now explain how the tracing experiment for Construction 8.1 leads to a QHPS relative to $(\mathcal{Q}^{\mathsf{Prod}}, \alpha^{\mathsf{Prod}}, \Omega_\delta^{\mathsf{Prod}}, R_\delta^{\mathsf{Prod}})$. Given an adversary $\mathcal{A}^{\mathsf{TT}}$, we define the QHPS $\mathcal{S}^{\mathsf{Prod}}$ as follows: first run the tracing experiment with $\mathcal{A}^{\mathsf{TT}}$, until $\mathcal{A}^{\mathsf{TT}}$ outputs $(|\mathsf{D}\rangle, m_0^*, m_1^*\rangle)$. Then set $|P\rangle = |\mathsf{D}\rangle$. Let S be the set of id queried by $\mathcal{A}^{\mathsf{TT}}$, and let S' be the corresponding set of codewords in the IPP code given to the users in S. Let Π be the valid partition generated by S'.

Finally, let $\mathcal{D}(q)$ be the algorithm which does the following: choose a random subset $T \subseteq [\ell]$ of size t. Choose a random bit b, and let $m = m_b^*$. Then let $m_j, j \in T$ be a t-out-of-t secret sharing of m: m_j are uniform conditioned on $\oplus_{j\in T} m_j = m$. For each $j \in T, \sigma \in \Sigma$, compute

$$c_{j,\sigma} = \begin{cases} \mathsf{Enc}^{\mathsf{PK}}(\mathsf{ek}_{j,\sigma}, m_j) & \text{if } \sigma \leq q_j \\ \mathsf{Enc}^{\mathsf{PK}}(\mathsf{ek}_{j,\sigma}, 0) & \text{otherwise} \end{cases}$$

Output $c = (T, \{c_{j,\sigma}\}_{j\in T, \sigma\in\Sigma})$ and b. In other words, $\mathcal{D}(q)$ outputs an encryption of a random choice of m_b^*, except that it replaces the ciphertext components $c_{j,\sigma}$ with junk (importantly, independent of b) if $q_j < \sigma$.

The QHPS then outputs $(|P\rangle, \Pi, \mathcal{D})$.

Lemma 8.2. *Assuming Π^{PK} is semantically secure and either (1) $t \geq (1-\delta)\ell$ or (2) $\binom{(1-\delta)\ell}{t}/\binom{\ell}{t}$ is negligible, then $\mathcal{S}^{\mathsf{Prod}}$ is valid for $\mathcal{Q}^{\mathsf{Prod}}, \alpha^{\mathsf{Prod}}, \Omega^{\mathsf{Prod}}, R^{\mathsf{Prod}}$.*

Proof. Let $q \in \Omega^{\mathsf{Prod}}$. Then q has strictly more than $\delta\ell$ zeros; call this set $\mathcal{Z}$. If $T \cap \mathcal{Z} \neq \emptyset$, then the share m_j inside $T \cap \mathcal{Z}$ is information-theoretically hidden given c. Since the m_j form a t-out-of-t secret sharing, this means if $T \cap \mathcal{Z} \neq \emptyset$, then b is statistically hidden. Thus, to prove the sink indistinguishability problem is hard, we just need to show that $T \cap \mathcal{Z} \neq \emptyset$ with overwhelming probability. A simple combinatorial argument shows that

$$\begin{aligned} \Pr[T \cap \mathcal{Z} \neq \emptyset] = 1 - \Pr[T \cap \mathcal{Z} = \emptyset] &= 1 - \frac{\binom{\ell - |\mathcal{Z}|}{t}}{\binom{\ell}{t}} \geq 1 - \frac{\binom{(1-\delta)\ell - 1}{t}}{\binom{\ell}{t}} \\ &= 1 - \left(\frac{(1-\delta)\ell - t}{(1-\delta)\ell}\right)\left(\frac{\binom{(1-\delta)\ell}{t}}{\binom{\ell}{t}}\right) \end{aligned}$$

If $t > (1-\delta)\ell$, then $\binom{(1-\delta)\ell}{t} = 0$. If $t = (1-\delta)\ell$, then $(1-\delta)\ell - t = 0$. In either case, $\Pr[T \cap \mathcal{Z} \neq \emptyset] = 1$. Alternatively, if $\binom{(1-\delta)\ell}{t}/\binom{\ell}{t}$ is negligible, then $\Pr[T \cap \mathcal{Z} \neq \emptyset] \geq 1 - \mathsf{negl}$. Thus, under the conditions of Lemma 8.2, the sink indisitnguishability problem is hard.

We now turn to the partition indisitnguishability experiment. Suppose q_1, q_2 are in the same part of Π, and consider the distributions $\mathcal{D}(q_1)$ and $\mathcal{D}(q_2)$. We will argue they are indistinguishable. Toward that end, consider some $j \in [\ell]$. Since q_1, q_2 are in the same part of Π, then $q_{1,j}$ and $q_{2,j}$ are in the same part of Π_j. If we assume wlog that $q_{1,j} \leq q_{2,j}$, then Π_i has no boundary in $(q_{1,j}, q_{2,j}]$. Now consider a ciphertext component $c_{j,\sigma}$. If $\sigma \leq q_{1,j}$, then the distributions of $c_{j,\sigma}$ under $\mathcal{D}(q_1)$ and $\mathcal{D}(q_2)$ are identical. Likewise if $\sigma > q_{2,j}$. On the other hand, for $\sigma \in (q_{1,j}, q_{2,j}]$, $\mathcal{A}^{\mathsf{TT}}$ does not have the secret key $\mathsf{dk}_{j,\sigma}$. By the semantic security of Π^{PK}, the adversary cannot distinguish $c_{j,\sigma}$ under $\mathcal{D}(q_1)$ and $\mathcal{D}(q_2)$.

By a hybrid over all j, no efficient adversary can distinguish $\mathcal{D}(q_1)$ from $\mathcal{D}(q_2)$, proving the hardness of the partition indistinguishability experiment. □

8.2 The BOS Algorithm

Algorithm 8.3 (BOS Algorithm $\mathcal{A}^{\mathsf{Prod}}$)**.** Initialize $x \in [0, s]^\ell$, and set $x = s^\ell$. Now query on x, which by definition gives 1. Then do the following for $i = 1, \ldots, \ell + 1$:

1. For $a = s, \ldots, 1$:
   (a) Set $x' = x$, except that x'_i is set to $a - 1$.
   (b) Query on x', receiving response o'.
   (c) If $o' = 0$, query on x, break the loop in Step 1, and proceed to the next iteration of the main loop over i.
   (d) Otherwise, set $x = x'$, and proceed to the next iteration of the loop over a in Step 1.
2. If the loop in Step 1 terminated with all responses o' being 1, then set $x_i = 0$. Then proceed to the next iteration of the main loop over i.

In the end, output w, which is x but with every 0 replaced by ?.

Theorem 8.4. *$\mathcal{A}^{\mathsf{Prod}}$ solves $(\mathcal{Q}^{\mathsf{Prod}}, \alpha^{\mathsf{Prod}}, \Omega_\delta^{\mathsf{Prod}}, R_\delta^{\mathsf{Prod}})$ in the BOS model.*

Proof. We maintain the invariant that each time we exit an iteration of the main loop over i, the last query was on x and the response was 1. There are two cases:

- We exited the iteration from Step 1c. But in this step we query on x, so x is the most recent query at exit. Moreover, x was queried in the last iteration of the loop over a in Step 1, and the result must have been 1 in order to proceed to this iteration. Therefore, x was queried two queries ago and resulted in response 1. By single step rewinding, the latest query on x must also give 1.
- We existed the iteration from Step 2. But here the most recent query was on x', it resulted in query response 1, and in Step 1d we set $x = x'$.

We also see that if we exit the loop of Step 1 via Step 1c, then the adjacent queries x and x' resulted in different outcomes, meaning a is a boundary of Π_i.

The result is that the final x is a string where all the non-zero terms are boundaries of the respective component partition, and $x \notin \Omega^{\mathsf{Prod}}$ (since the query output was no 0), meaning the number of 0's in x is at most $\delta\ell$. Thus when we replace 0's with ?'s to get w, we have that $R^{\mathsf{Prod}}(\Pi, w) = 1$. We finally note that the number of queries $\mathcal{A}^{\mathsf{Prod}}$ makes is at most $O(s\ell)$, which is polynomial. □

Corollary 8.5. *Assuming Π^{PK} is a secure PKE scheme, Construction 8.1 is quantum traceable.*

9 Tracing Embedded Identities with Short Ciphertexts

In Construction 7.1 from Section 7, the string x is n bits long, where n is the bit length of identities. It is also part of the message inputted to $\mathsf{Enc}^{\mathsf{FE}}$ of the underlying functional encryption scheme. Therefore, the ciphertext size must grow with the bit length of identities. As observed by [NWZ16], this is *not* inherent to traitor tracing. They instead propose a different structure where ciphertexts may be independent of the identity length. We now recall their construction:

Construction 9.1. Let Π^{FE} be a functional encryption scheme. We construct the traitor tracing scheme $\Pi^{\mathsf{TT}} = (\mathsf{Gen}^{\mathsf{TT}}, \mathsf{Derive}^{\mathsf{TT}}, \mathsf{Enc}^{\mathsf{TT}}, \mathsf{Dec}^{\mathsf{TT}}, \mathsf{Trace}^{\mathsf{TT}})$, where we define $\mathsf{Gen}^{\mathsf{TT}}, \mathsf{Derive}^{\mathsf{TT}}, \mathsf{Enc}^{\mathsf{TT}}, \mathsf{Dec}^{\mathsf{TT}}$ below:

- $\mathsf{Gen}^{\mathsf{TT}}(1^\lambda) = \mathsf{Gen}^{\mathsf{FE}}(1^\lambda)$
- $\mathsf{Derive}^{\mathsf{TT}}(\mathsf{msk}, \mathsf{id})$: choose a random $\tau \in [N]$ where $N = 2^\lambda$. Then return $\mathsf{sk}_{\tau,\mathsf{id}} \leftarrow \mathsf{Derive}^{\mathsf{FE}}(\mathsf{msk}, f_{\tau,\mathsf{id}})$ where $f_{\tau,\mathsf{id}}(i, x, m) = \begin{cases} m & \text{if } x \geq 2 * \tau - \mathsf{id}_i \\ \bot & \text{otherwise} \end{cases}$. Note here that $x \in [0, 2N] = [0, 2^{\lambda+1}]$ and $i \in [n]$, so the inputs to $\mathsf{Enc}^{\mathsf{TT}}$ are bounded by $O(\lambda)$ bits long, independent of n. Thus if Π^{FE} has succinct ciphertexts, so will Π^{TT}.
- $\mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m) = \mathsf{Enc}^{\mathsf{FE}}(\mathsf{pk}, (1, N, m))$
- $\mathsf{Dec}^{\mathsf{TT}}(\mathsf{sk}_{\tau,\mathsf{id}}, c) = \mathsf{Dec}^{\mathsf{FE}}(\mathsf{sk}_{\tau,\mathsf{id}}, c)$

Tracing. The classical ideal behind the structure above is the following: first apply the tracing algorithm for Construction 7.1 setting $i = 1$. The result is that the tracing algorithm outputs $2\tau - \mathsf{id}_1$ from some secret key. Since id_1 is a single bit, this reveals uniquely both τ and the first bit of the associated identity. Then, by varying i and testing on ciphertexts encrypting $(i, 2\tau - 1, m)$, one can learn the rest of the bits of id. Essentially, we know if the first phase accused $2\tau - \mathsf{id}_1$, there must be a gap in the success probabilities of the decoder on encryptions of $(1, 2\tau - \mathsf{id}_1, m)$ and $(1, 2\tau - \mathsf{id}_1 - 1, m)$. Call these two probabilities p_0 and p_1, respectively, which are far apart. Since the τ of various secret keys are unique whp, by the security of functional encryption, we know that the decryption probability for $(i, 2\tau - \mathsf{id}_i, m)$ must be close to p_0, and the decryption probability for $(i, 2\tau - \mathsf{id}_i - 1, m)$ must be close to p_1. Put another way, the decryption probability for $(i, 2\tau - 1, m)$ will be close to $p_{1-\mathsf{id}_i}$, thus revealing id_i.

Quantumly, we can employ the same strategy to recover τ, id_1. However, recovering the rest of the bits of id will not work as in the classical case. This is because the probabilities p_0, p_1 may change as we further interrogate the decoder, so we cannot simply compare with the previous value. We therefore need to develop a new quantum algorithm for the problem.

9.1 The Hidden Partition

Let $Q^{\mathsf{Short}} = [n] \times [0, 2N]$ where $N = 2^\lambda$. Let $\alpha^{\mathsf{Short}} = (1, 2N)$. Let $\Omega^{\mathsf{Short}} = \{(i, 0)\}$. Consider a sequence $(\tau^{(1)}, \mathsf{id}^{(1)}), \ldots, (\tau^{(t)}, \mathsf{id}^{(t)})$ where $\mathsf{id}^{(j)} \in \{0,1\}^*$ and $\tau^{(j)} \in [N]$ with $\tau^{(1)} < \tau^{(2)} < \cdots < \tau^{(t)}$. This gives rise to a partition $\Pi = \{S_0, \ldots, S_t\}$ of Q^{Short} consisting of sets $S_j = \{(i, x) : 2\tau^{(j)} - \mathsf{id}_i^{(j)} \leq x < 2\tau^{(j+1)} - \mathsf{id}_i^{(j+1)}\}$ for $j = 1, \ldots, t-1$, $S_0 = \{(i, x) : x < 2\tau^{(1)} - \mathsf{id}_i^{(1)}\}$, and $S_t = \{(i, x) : 2\tau^{(t)} - \mathsf{id}_i^{(t)} \leq x\}$. We call Π of this form *contiguous*, and we call $(\tau^{(j)}, \mathsf{id}^{(j)})$ the boundaries of Π. Let $R^{\mathsf{Short}}(\Pi, w)$ be the following relation:

- Output 1 if Π is *not* valid
- Output 1 if Π is valid, and w is a boundary of Π.
- Output 0 if Π is valid but w is not a boundary of Π.

We now explain how Construction 9.1 leads to the quantum hidden partition problem relative to this partition. We interpret an adversary $\mathcal{A}^{\mathsf{TT}}$ interacting in the tracing experiment as a quantum hidden partition sampler $\mathcal{S}^{\mathsf{TT}}$, where $|P\rangle$ is the decoder $|\mathsf{D}\rangle$ outputted by $\mathcal{A}^{\mathsf{TT}}$, Π is the contiguous partition whose boundaries are the identities queried by $\mathcal{A}^{\mathsf{TT}}$. Finally $\mathcal{D}$, on input (i,q), encrypts $((i,q),m_b)$ for a random choice of b to get ciphertext c, and outputs (c,b).

Lemma 9.2. *Assuming Π^{FE} is a secure FE scheme, the QHPS is valid for $\mathcal{Q}^{\mathsf{Short}},\alpha^{\mathsf{Short}},\Omega^{\mathsf{Short}},R^{\mathsf{Short}}$*

Proof. The sinks have the form $(i,0)$, and note that encryptions of $((i,0),m)$ computationally hide m, since the only secret keys are f_{id} for $\mathsf{id}>0$. Thus the probability the adversary can predict b (and therefore win the sink indistinguishability experiment) is at most $1/2+\mathsf{negl}$.

Next, observe that encryptions of $((i,q),m)$ and $((i',q'),m)$ decrypt identically if (i,q) and (i',q') are in the same part of Π. Thus, by FE security, the encryptions are indistinguishable if they lie in the same part. Therefore, the probability of winning the partition indistinguishability experiment is $1/2+\mathsf{negl}$. □

9.2 The BOS Algorithm

Algorithm 9.3 (BOS Algorithm $\mathcal{A}^{\mathsf{Short}}$)**.** Initialize integers $a,b\in[0,2N]$ and $i\in[n]$. Set $a=2N,b=0$, and query on $(1,a)$, which by definition gives response $o=1$. Then do the following for at most $O(kn\log^2 N)$ steps, where k in an upper bound on the number of parts in Π:

1. Query on $(1,b)$, obtaining response o.
2. If $o=1$, set $(a,b)=(b,0)$ and go to Step 1.
3. Else $(o=0)$:
   (a) If $b\neq(a-1)$: Query on $(1,a)$, set $b=\lfloor(a+b)/2\rfloor$ (a remains unchanged), and go to Step 1.
   (b) Else $(b=a-1)$: Query on $(1,a)$, parse a as $2\tau-\mathsf{id}_1$, and do the following for $i=2,\ldots,n$:
      i. Query on $(i,2\tau-1)$, receiving response o.
      ii. If $o=0$, query on $(1,a)$, set $\mathsf{id}_i=0$, and proceed to the next i.
      iii. Else $(o=1)$:
         A. Query on $(i,2\tau-2)$, receiving response o'.
         B. If $o'=0$, set $\mathsf{id}_i=1$, query on $(i,2\tau-1)$ and then $(1,a)$ and proceed to the next i.
         C. Else $(o'=1)$, set $(a,b)=(a-1,0)$, clear τ,id, exit the loop over i in Step 3b, and go to Step 1.

   If the loop over i in Step 3b completes (that is, if we never have $o'=1$), output (τ,id).

Theorem 9.4. *$\mathcal{A}^{\mathsf{Short}}$ solves $(\mathcal{Q}^{\mathsf{Short}},\alpha^{\mathsf{Short}},\Omega^{\mathsf{Short}},R^{\mathsf{Short}})$ in the BOS model.*

Proof. The proof is an adaptation of the proof of Theorem 6.3 for $\mathcal{A}^{\mathsf{Bnd}}$. As in Theorem 6.3, we will eventually find $a,b=a-1$ where the previous query was on $(1,a-1)$ and gave 0, and the query before was on $(1,a)$ and gave 1. This is Step 3b. By local consistency, $a=2\tau-\mathsf{id}_1$ for some (τ,id) that defined the partition. We then query on $(1,a)$ again, which by single step rewinding gives 1.

It remains to show that id is obtained in its entirety. This is done by trying to ascertain if $(i, 2\tau - 1)$ lies in the same part as $(1, a)$ or $(1, a - 1)$. This is done by querying on $(i, 2\tau - 1)$. If the result is 0 (Step 3(b)ii), we know by local consistency (and the fact that the last query was on $(1, a)$ and gave 1) that $(i, 2\tau - 1)$ cannot be in the same part as $(1, a)$. So we set $\mathsf{id}_i = 0$ and move on to the next bit of id. In order to guarantee that the last query response was 1, we query again on $(1, a)$, which, by single-step rewinding will give 1.

If $o = 1$, we do not immediately learn which part $(i, 2\tau - 1)$ is in. So we also query on $(i, 2\tau - 2)$ to get response o' (Step 3(b)iiiA). If the response is now 0, we know that $(i, 2\tau - 1)$ and $(i, 2\tau - 2)$ are in different parts, meaning $(i, 2\tau - 1)$ is in the same part as $(1, a)$. Thus we set $\mathsf{id}_i = 1$ and move on to the next bit of id (Step 3(b)iiiB). But first we query again on $(i, 2\tau - 1)$, which by single-step rewinding gives 1. Then we query on $(1, a)$, which gives 1 by local consistency.

If on the other hand the response o' is once again 1, then we learn nothing about id_i. However, in this case, we have exactly the setup of a new hidden partition with $N = \tau - 1$. In particular, we know there must be a (τ', id') with $\tau' < \tau$. We therefore proceed back to Step 1 but with the new $N = \tau - 1$. A simple inductive argument shows that we will eventually find such a (τ', id'). This completes the proof. □

Corollary 9.5. *Assuming FE, Construction 9.1 is quantum traceable.*

Proof. Let $\mathcal{T}(|P\rangle, 1^{1/\epsilon})$ be the algorithm guaranteed by Theorem 5.5 and the existence of $\mathcal{A}^{\mathsf{Short}}$. Then set $\mathsf{Trace}^{\mathsf{TT}}(\mathsf{pk}, m_0, m_1, 1^{1/\epsilon}, |\mathsf{D}\rangle)$ to be $\mathcal{T}^{\mathcal{D}}(|\mathsf{D}\rangle, 1^{1/\epsilon})$ where $\mathcal{D}$ is the sampler obtained from pk, m_0, m_1 described in Section 9.1. Now consider the events $\mathsf{Live}_\epsilon^{\mathsf{BOS}}, \mathsf{Success}_\epsilon^{\mathsf{BOS}}, \mathsf{Fail}_\epsilon^{\mathsf{BOS}}$ for the QHPS described in Section 9.1, and the events $\mathsf{Live}_\epsilon^{\mathsf{TT}}, \mathsf{Success}_\epsilon^{\mathsf{TT}}, \mathsf{Fail}_\epsilon^{\mathsf{TT}}$ for the tracing experiment with Π^{TT} from Construction 9.1; in particular $D(\alpha)$ is identical to running $\mathsf{Enc}^{\mathsf{TT}}(\mathsf{pk}, m_b)$ for a random choice of b. We see that the events exactly coincide. Thus, by the guarantees of Theorem 5.5, Π^{TT} in Construction 9.1 is secure. □

10 Barriers to Generalizing our Results

A natural question is: what are the limits of the BOS model and our realization using quantum rewinding? After all, Zhandry's prior work [Zha20] and our results in Sections 6, 7, 8, and 9 show that the all the hidden partition problems arising from existing literature (that we are aware of) can be handled quantumly. So perhaps it may be possible to solve *all* quantum hidden partition problems?

There are at least two potential routes toward a generalization. The first is to extend our results of this section to show that quantum rewinding can be used to realize a generalization of the BOS model (Definition 5.3) with single-step rewinding replaced by k-step rewinding: for any $i, j \leq k$, if $q_{i+1+j} = q_i$, then $o_{i+1+j} = o_i$. Current techniques, unfortunately, are incapable of extending beyond the $k = 1$ we consider in this work. This is because current techniques all rely on Jordan's lemma, which characterizes how two projectors interact. To move to even $k = 2$, one would need to characterize how three projectors interact (more generally, $k + 1$ projectors), and there does not appear any analog of Jordan's lemma beyond two projectors.

The second route toward generalization is to show that the BOS model is equivalent to the Global Hidden Partition model, in that any hidden partition instance that can be solved in the global model can be solved in the BOS model as well.

Here, we rule out this second route. Specifically, we give an example of a hidden partition problem that is solvable if we generalize the BOS model to have even 2-step rewinding, but unsolvable with 1-step rewinding:

Theorem 10.1. *There exists an $R, \mathcal{Q}, \alpha, \Omega$ that is solvable with 2-step rewinding (with probability 1), but that the probability it can be solved in the BOS model is negligible.*

Proof. Our hidden partition problem will consist of two copies of the PLBE hidden partition from Section 6, glued together at N. Specifically, $\mathcal{Q}^{\mathsf{Bnd2}}$ will be set to be $\{0, 0', 1, 1', \ldots, N-1, (N-1)', N\}$. A partition Π will then look like $\{[0, a_1-1], [0', a_1'-1], [a_1, a_2-1], [a_1', a_2'-1] \ldots, [a_{k-1}, a_k-1], [a_{k-1}', a_k'-1], [a_t, N] \cup [a_t', N]\}$. The sinks are $0, 0'$, the source is N (which equals N'). Our goal is to output one of the a_i *and* one of the a_i'. We will have N be super-polynomial.

We can easily accomplish this with 2-step rewinding. We first trace the un-primed version copy to recover an a_i, using our algorithm for $\mathcal{A}^{\mathsf{Bnd}}$ (Algorithm 6.2) except that we modify the query sequence in order to make sure that there is always a relatively recent query to N that gave 1. To do so, we present the $\mathcal{A}^{\mathsf{Bnd}}$ with the oracle O', which answers each query q_i as follows:

- Make a query to O on N.
- If the query made two to O' two queries ago, namely q_{i-2}, was *not* equal to q_i, then make a query to O on q_{i-1}.
- Finally, regardless of if $q_{i-2} = q_i$ or not, make a query to O on q_i. Return as the output of O' whatever O outputs.

In other words, we map the query sequence $N, q_1, q_2, q_3, q_4, q_5, \ldots$ to O' to the sequence of queries $N, q_1, q_2, N, q_2, q_3, N, q_3, q_4, N, q_4, q_5, \ldots$, except that we delete the second query to q_{i-1} if $q_i = q_{i-2}$.

Observe that all queries to N have at most two intermediate queries. Thus 2-step rewinding on O guarantees that all queries made to O on N result in 1. Moreover, if $q_i = q_{i-2}$, then q_{i-2} was queried to O exactly two queries ago, and so 2-step rewinding guarantees that O (and hence O') gives the same output on q_i as it did on q_{i-2}, hence guaranteeing that O' satisfies 1-step rewinding. Finally, local consistency for O and the fact that the first query to q_i is immediately preceded by the second query to q_{i-1} guaranteed that if q_i and q_{i-1} are in the same part, then the response to the first query on q_i is the same as the response on the second query to q_{i-1} (except in cases where 1-step rewinding kicks in). Moreover, 2-step rewinding on O means that the first and second queries to q_{i-1} give the same output (since they are spaced 1 query apart). This implies that O' has 1-step rewinding.

Thus O' is a BOS oracle, and by Theorem 6.3, $\mathcal{A}^{\mathsf{Bnd}}$ thus outputs an a_i. But now we can trace the primed copy using $\mathcal{A}^{\mathsf{Bnd}}$. We start with a query to N, which is guaranteed to give $o = 1$ since there was a recent query to N giving 1. So tracing via Theorem 6.3 on the primed side will succeed and give us an a_i'.

On the other hand, with 1-step rewinding we can guarantee failure.

Consider sampling a_1, a_1' uniformly in $[N/4, N/2-1]$, and then a_2, a_2' uniformly in $[N/2+1, 3N/4]$. We then consider the following BOS oracle O:

- O starts out in Mode 1. It stays in Mode 1 until it sees two consecutive queries that are both not in $[a_2, N] \cup [a_2', N]$. In Mode 1, it answers all queries in $[N/2+1, N] \cup [(N/2+1)', N]$ with 1, and all queries in $[0, N/2] \cup [0', (N/2)']$ with 0.

- On any query q_i, if both the current query q_i and previous query q_{i-1} were not in $[a_2, N] \cup [a_2', N]$, then O switches permanently to one of the following three modes:
  - If $q_{i-1} \in [N/2+1, a_2 - 1]$ then switch to Mode 2.1. In Mode 2.1, all queries in $[a_1, a_2 - 1]$ are answered with 1, and all other queries are answered with 0. Answer the query q_i as in Mode 2.1.
  - If $q_{i-1} \in [(N/2+1)', a_2' - 1]$ then switch to Mode 2.1'. In Mode 2.1', all queries in $[a_1', a_2' - 1]$ are answered with 1, and all other queries are answered with 0. Answer the query q_i as in Mode 2.1'.
  - If $q_{i-1} \in [0, N/2] \cup [0', (N/2)']$ then switch to Mode 2.0. In Mode 2.0, all queries are answered with 0. Answer the query q_i as in Mode 2.0.

Lemma 10.2. *O is a BOS algorithm.*

Proof. Indeed Mode 1, any queries outside $[a_2, N] \cup [a_2', N]$ have at least a query between them, meaning local consistency is vacuous for these queries. Local consistency is also enforced on $[a_2, N] \cup [a_2', N]$ by the fact that Mode 1 always responds with 1. 1-step rewinding in turn is guaranteed by the fact that the oracle in Mode 1 is stateless. Modes 2.0,2.1,2.1' are globally consistent, so they satisfy local consistency and 1-step rewinding as well.

It then remains to check the transition out of Mode 1. We first observe that if making the transition, the three most recent queries were (in reverse order):

- $q_i \notin [a_2, N] \cup [a_2', N]$ (by definition)
- $q_{i-1} \notin [a_2, N] \cup [a_2', N]$ (by definition).
- $q_{i-2} \in [a_2, N] \cup [a_2', N]$ (since we did not transition at query q_{i-1}).

In particular, 1-step rewinding is not relevant for query q_i. Then the Modes 2.0,2.1,2.1' are chosen to ensure local consistency for q_i, and global consistency for q_{i+1}. For q_{i+2} ad beyond, the transition is complete and the global consistency of Modes 2.0,2.1,2.1' kick in. □

Now we observe that O contains no information about the a_i or a_i' in Mode 1. Moreover, All the Modes 2.x contain information about either the a_1, a_2, or a_1', a_2' (or neither), but not both. Therefore, the probability of outputting both a primed and non-primed a_i is at most $O(1/N)$, which is negligible. □

Remark 10.3. Note that if N is polynomial, the proof above only gives an inverse-polynomial bound on the success probability. We can extend the proof of Theorem 10.1 to handle polynomial N but exponential success probability by glueing together a polynomial number of copied of the PLBE hidden partition, each copy being polynomial size. The required output will then be a boundary from *every* one of the copies.

References

[ABG+13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. Cryptology ePrint Archive, Report 2013/689, 2013. https://eprint.iacr.org/2013/689.

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

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, February 2014.

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

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

[BN08] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 501–510. ACM Press, October 2008.

[BP08] Olivier Billet and Duong Hieu Phan. Efficient traitor tracing from collusion secure codes. In Reihaneh Safavi-Naini, editor, *ICITS 08*, volume 5155 of *LNCS*, pages 171–182. Springer, Heidelberg, August 2008.

[BS95] Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465. Springer, Heidelberg, August 1995.

[BSW06] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Heidelberg, May / June 2006.

[CFN94] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Heidelberg, August 1994.

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 556–584, Virtual Event, August 2021. Springer, Heidelberg.

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments. In *Proceedings of FOCS 2021*, 2021.

[DFNS14] Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. Superposition attacks on cryptographic protocols. In Carles Padró, editor, *ICITS 13*, volume 8317 of *LNCS*, pages 142–161. Springer, Heidelberg, 2014.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018.

[GKW19] Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 149–179. Springer, Heidelberg, December 2019.

[GZ20] Marios Georgiou and Mark Zhandry. Unclonable decryption keys. Cryptology ePrint Archive, Report 2020/877, 2020. https://eprint.iacr.org/2020/877.

[HvLT98] Henk D.L Hollmann, Jack H van Lint, Jean-Paul Linnartz, and Ludo M.G.M Tolhuizen. On codes with the identifiable parent property. *Journal of Combinatorial Theory, Series A*, 82(2):121–133, 1998.

[KM10] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685, 2010.

[KN22] Fuyuki Kitagawa and Ryo Nishimaki. Watermarking PRFs against quantum adversaries. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part III*, volume 13277 of *LNCS*, pages 488–518. Springer, Heidelberg, May / June 2022.

[LMS21] Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited (or: How to do quantum rewinding undetectably). Cryptology ePrint Archive, Report 2021/1543, 2021. https://eprint.iacr.org/2021/1543.

[MW04] C. Marriott and J. Watrous. Quantum arthur-merlin games. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 275–285, 2004.

[NWZ16] Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 388–419. Springer, Heidelberg, May 2016.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

[Sir06] Thomas Sirvent. Traitor tracing scheme with constant ciphertext rate against powerful pirates. Cryptology ePrint Archive, Report 2006/383, 2006. https://eprint.iacr.org/2006/383.

[Tar03] Gábor Tardos. Optimal probabilistic fingerprint codes. In *35th ACM STOC*, pages 116–125. ACM Press, June 2003.

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, Heidelberg, April 2012.

[VDG98] Jeroen Van De Graaf. *Towards a Formal Definition of Security for Quantum Protocols*. PhD thesis, Universite de Montreal, 1998.

[Wat06] John Watrous. Zero-knowledge against quantum attacks. In Jon M. Kleinberg, editor, *38th ACM STOC*, pages 296–305. ACM Press, May 2006.

[Zha12] Mark Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, October 2012.

[Zha20] Mark Zhandry. Schrödinger's pirate: How to trace a quantum decoder. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 61–91. Springer, Heidelberg, November 2020.