Schrödinger's Pirate: How To Trace a Quantum Decoder

Mark Zhandry
Princeton University & NTT Research
mzhandry@gmail.com

Abstract

We explore the problem of traitor tracing where the pirate decoder can contain a quantum state. Our main results include:

- We show how to overcome numerous definitional challenges to give a meaningful notion of tracing for quantum decoders
- We give negative results, demonstrating barriers to adapting classical tracing algorithms to the quantum decoder setting.
- On the other hand, we show how to trace quantum decoders in the setting of (public key) private linear broadcast encryption, capturing a common approach to traitor tracing.

1 Introduction

Quantum computers pose a looming threat to cryptography. By an unfortunate coincidence, the enhanced computational power of quantum computers allows for solving the exact mathematical problems, such as factoring and discrete log, underlying the bulk of public-key cryptography used today [Sho94]. The good news is that "quantum-safe" mathematical tools—such as lattices, multivariate equations, or isogenies—exist that can be used as a drop-in replacement in many setting. Nevertheless, many challenges remain. For example, using a quantum-safe drop-in replacement does not always guarantee the security of the overall protocol, as many of the classical proof techniques fail to carry over to the quantum setting [VDG98, ARU14, BDF+11]. It may also be that quantum attackers may get "superposition access" to the honest parties, opening up new avenues of attack [KM10, Zha12a, DFNS14, KLLN16].

In this work, we consider an entirely different threat from quantum computers, which to our knowledge has not been identified before: quantum piracy!

Traitor Tracing. The focus of this work will be the setting of traitor tracing, one of the fundamental goals in cryptography. Originally defined by Chor, Fiat and Naor [CFN94], traitor tracing helps protect content distributors from piracy. In such a system, every legitimate user has their own secret decryption key which can decrypt ciphertexts. The content distributor is worried about a user distributing their key to unauthorized users. Of course, little can be done to stop a user from distributing their key. Instead, in the event that the distributor discovers an unauthorized decryption key, the distributor would like to identify the source of the key, so that the user (deemed a "traitor") can be prosecuted or have their credentials revoked. This "tracing" should be possible even if the user tries to hide their identity, say, by embedding their key in an obfuscated pirate decoder

program. What's more, tracing should still succeed even if many malicious users pool their keys into a single decoder. As sketched in [CFN94], classical tracing can readily be build from generic public key encryption, albeit with large ciphertexts. Therefore, the goal is typically to devise traitor tracing with small ciphertexts. Numerous number-theoretic [BSW06, GGH+13, BZ14, GKW18] and combinatorial schemes [CFN94, BN08] have been shown, with various trade-offs between system parameters and the computational assumptions needed for security.

Most of cryptography concerns several honest parties communicating with each other, while an adversary eavesdrops or manipulates the communication between them. Traitor tracing is in some sense the opposite: several *dis*honest parties (namely, the traitor(s) and the receiver of the pirate decoder) communicate, while the *honest* party (the content distributor) is intercepting this communication (the decoder). This role reversal makes traitor tracing a fascinating problem, as the very cryptographic techniques employed to help secure communication between honest partiescan be employed by the dishonest parties in an attempt to hide their identity and protect themselves from being traced.

Traitor Tracing Meets Quantum Attackers. The aforementioned role reversal also has interesting consequences once quantum computers are involved, as we now highlight. Certainly, the underlying mathematical tools now need to be quantum resistant; for example, post-quantum obfuscation [BGMZ18] or LWE-based traitor tracing [GKW18] can be used. The proofs of security must also work for quantum attackers; existing traitor tracing schemes satisfy this as well. What is obtained is the following: if a *classical* pirate decoder is intercepted from a quantum traitor, that traitor can be identified.

But now suppose the traitor has a quantum computer and is sending its decoder to a quantum recipient. Just as a classical traitor can attempt to use classical cryptographic techniques to evade detection, this quantum traitor could now try to leverage *quantum cryptography*. Quantum cryptography uses the unusual features of quantum physics such as no-cloning to achieve never-before-possible applications, such as information-theoretic key agreement [BB87], unforgeable currency [Wie83, AC12], unclonable programs [Aar09], certifiable randomness [BCM+18], and secret keys that self-destruct after use [AGKZ20].

Therefore, we can imagine the traitors creating and sending a decoder comprising a *quantum* state. We stress that the entire system remains classical under normal operation: keys, ciphertexts, encryption, and decryption are all entirely classical and can be run on classical computers and classical networks. The attacker only ever receives classical communication from the honest parties. Even so, the quantum attackers can use a communication channel outside of the system: they can meet in person to exchange the decoder, or perhaps send the decoder over an outside quantum-enabled network. Nothing the content distributor does can prevent the traitor from sending a quantum decoding device.

Existing traitor tracing results *do not* handle such quantum decoders. In more detail, essentially all classical tracing algorithms work by testing a decoder on a variety of different ciphertexts and examining the outputs. When moving to quantum decoders, the measurement principle in quantum mechanics means that extracting information from a quantum state may irreversibly alter it. This means, after potentially the first ciphertext is decrypted, the decoder's state may be irreversibly altered into a state that is no longer capable of decrypting, essentially self-destructing. Now, a useful pirate decoder would likely *not* self-destruct on valid ciphertexts. However, a decoder that eventually self-destructs but evades tracing may be a worthwhile compromise for a traitor. Moreover,

all classical tracing algorithms will also run the decoder on many *invalid* ciphertexts, and the utility of the decoder does not require it to decrypt such ciphertexts.

The above discussion means even the most basic of classical traitor tracing results—for example, the aforementioned generic scheme from public key encryption—may no longer work in the setting of quantum decoders. In fact, it turns out that even *defining* tracing in this setting is non-trivial, for reasons discussed in Section 1.2 below.

We note that similar issues may arise any time there is adversarial communication that the *honest* party is trying to learn information from. In such cases, the adversary may benefit from using quantum communication, even if the cryptosystem itself entirely classical. Software watermarking [BGI+01, CHN+16] is another example of where such issues may arise. In such cases, classical security proofs should be revisited, and new techniques are likely needed. In this work, we focus exclusively on the case of traitor tracing, but we expect the tools we develop to be useful for other similar settings.

1.1 Our Results

Definition. Our first result is a new definition for what it means to be a secure tracing scheme in the presence of quantum decoders. As we will see, the obvious "quantumization" of the classical definition leads to a nonsensical definition. We must therefore carefully devise a correct quantum definition of traitor tracing, which requires developing new ideas.

Negative Result. One could have hoped that the tracing algorithm could be entirely classical, except for the part where it runs the decoder. We show barriers to such classical tracing algorithms, in particular showing that such algorithms cannot trace according to our security notion. Thus, any tracing algorithm satisfying our definition must be inherently quantum.

Positive Result. Finally, we develop a quantum tracing algorithm. Our tracing algorithm works on any private linear broadcast encryption (PLBE) scheme satisfying certain requirements. This in particular captures the constructions from generic public key encryption and from obfuscation, simply replacing the classical tracing algorithm with ours. As demonstrated by our negative result, our tracing requires new inherently quantum ideas. In particular, we employ a technique of [MW04], which was previously used in the entirely different setting of quantum Arthur-Merlin games.

1.2 Technical Overview

1.2.1 Live Quantum Decoders

For simplicity in the following discussion, we will assume the message space is just a single bit. Classically, the definition of security for a traitor tracing system is roughly as follows: define a "good" pirate decoder as one that can guess the message with probability noticeably larger than 1/2. Then security requires that any good pirate decoder can be traced with almost certainty to some user identity controlled by the adversary.

First, we will change terminology slightly. For a classical decoder, whether the decoder is good or bad is a fixed and immutable property. However, quantumly, whether the decoder can decrypt or not is potentially in flux as we disrupt the decoder by interrogating it. Therefore, we prefer the terms "live" and "dead" to "good" and "bad": a live decoder is one that, in its current state, would

successfully decrypt a random ciphertext. Unlike the classical case, a live decoder may become dead after such decryption.

We now describe several examples which illustrate the difficulties in defining liveness of quantum decoders.

Example 1. We will consider two simple attacks. In both cases, the adversary controls a single secret key sk_i for user i. It creates two programs, D_0 which has sk_i hard-coded and decrypts according to the honest decryption procedure, and D_1 which simply outputs a random bit.

The first adversary, A, chooses a random bit b, and outputs the decoder D_b. A is entirely classical, and any reasonable notion of liveness would assign D_0 to be live and D_1 to be dead, so A outputs a live decoder with probability $1/2$.

The second adversary, B, chooses a random bit b, and outputs the decoder

$$|☠_b\rangle = \frac{1}{\sqrt{2}}|D_0\rangle + \frac{(-1)^b}{\sqrt{2}}|D_1\rangle \ .$$

Here, $|☠_b\rangle$ is a quantum superposition of the two decoders D_0, D_1, with a "phase" that depends on b. To run the decoders, simply run in superposition to get the superposition of outputs of the decoders, finally measuring and outputting the result. The question is then: with what probability does B output a live decoder?

On one hand, we might be tempted to assign both decoders $|☠_0\rangle, |☠_1\rangle$ to be live, since both decoders can readily be verified to have a probability $3/4 > 1/2$ of decrypting. In any case, the phase does not fundamentally change the nature of the decoders, so any reasonable notion of liveness should assign $|☠_0\rangle$ and $|☠_1\rangle$ either both live or both dead. In this case, B's output is deterministically either live or dead. In particular, A and B have different distributions of liveness.

On the other hand, consider the *density matrices* of the outputs of A and B. For a quantum process outputting state $|\psi_i\rangle$ with probability p_i, the density matrix is $\sum_i p_i|\psi_i\rangle\langle\psi_i|$. According to the postulates of quantum mechanics, no physical operation (even computationally unbounded) can distinguish states with identical density matrices. But a routine calculation shows that the density matrices of A and B are in fact identical, meaning that the notion of liveness must be non-physical! Such a non-physical security definition cannot possibly reflect real-world security goals. We note that this example can be readily generalized to any non-trivial[1] way to assign liveness to quantum states.

Idea 1: Measuring the Decoder. We observe that $|☠_b\rangle$ in the above example are really just simple quantum versions of probability distributions: the decoders $|☠_b\rangle$ can be roughly thought of as being D_0 with probability $1/2$ and D_1 with probability $1/2$. For classical pirate decoders, similar issues arise if we try to apply the notion of "live" to the entire probability distribution over decoders. Instead, classically we would only consider the goodness of actual concrete pirate decoder produced by the adversary. The only thing quantum about our example is that it turned a probability distribution—which models uncertainty in the outcome, and is therefore non-physical—into a well-defined physical object.

[1] By non-trivial, we mean there is at least one live state and one dead state.

Motivated by the role of measurements in quantum mechanics, the natural solution to the above example is to consider $|\text{☠}_b\rangle$ as being a superposition over live and dead decoders[2]. The security definition and challenger will then *measure* whether the decoder is live or dead, rather than try to assign liveness to the overall quantum state. In the example above, this is done by simply measuring $|\text{☠}_b\rangle$, obtaining a random choice of D_0, D_1, and then performing the classical test for liveness. If the decoder measures to be live, then we require the decoder to actually be live, and moreover we require tracing to succeed. This easily resolves the above example, since measuring live vs dead will simply collapse the quantum decoder to a classical probability distribution.

More abstractly, a decoder has some actual probability $\hat{p}$ of decrypting random ciphertexts; in our $|\text{☠}_b\rangle$ example, $\hat{p} = 3/4$. However, this probability is hidden inside the quantum state and cannot be accessed in a physically meaningful way. The solution is instead to *measure* or *observe* the success probability, resulting in a measured success probability p. For $|\text{☠}_b\rangle$ as given above, when we observe p, we find that it can be either $1/2$ or 1, each with 50% probability.

Example 2. In the case of more general decoders, however, defining the procedure to measure success probabilities is non-trivial. We cannot in general simply perform the standard measurement as above, as doing so might break the decoder. As a simple example, the decoder's state could be the quantum Fourier transform applied to $|\text{☠}_b\rangle$ from the example above. Evaluation simply applies the inverse transform, recovering $|\text{☠}_b\rangle$, and then running the decoder as above. If we try to observe p by performing a standard measurement on this "encoded" decoder, the measurement will result in garbage. The observed p will therefore be $1/2$, despite the actual overall success probability of the decoder still being $3/4$.

In Example 2, we could of course define our measurement for p as: perform the inverse Fourier transform, and then perform the standard measurement. While this works for this particular case, the example illustrates that care is needed in determining how to measure liveness, and that the exact way we measure p will depend on the decoder itself. We need an automated way to determine the appropriate measurement that works, regardless of how $|\text{☠}\rangle$ operates.

Example 3. In the classical setting, the goodness or liveness of a decoder is determined by deciding whether the probability that the decoder correctly decrypts is above a given threshold. However, the exact probability cannot be computed efficiently: it amounts to determining the precise number of accepting inputs of a circuit, which is NP-hard. Therefore, most definitions of classical tracing are actually inefficient, in the sense that determining whether or not an adversary broke the security experiment cannot be determined in polynomial time.

Now, one could imagine *estimating* the success probability by simply running the decoder on several random ciphertexts. This gives rise to a definition of liveness that actually *can* be meaningfully translated to the quantum setting: namely, to measure liveness, run the decoder on several random ciphertexts in sequence, compute the fraction of ciphertexts that were correctly decrypted, and finally outputting "live" if the fraction exceeded a given threshold.

On the other hand, this notion of liveness has some limitations. First, suppose the measurement used q ciphertexts. Then the decoder could potentially decrypt q ciphertexts correctly and self-destruct. The decoder would measure as live, but actually result in a dead decoder, which would subsequently be untraceable.

[2]In much the same way that Schrödinger's cat is neither live nor dead, but is rather a superposition over live and dead cats.

Another issue is that this attempted notion of liveness is rather weak. A decoder may start off with a very high probability of decryption, and then reverse to a high probability of failure, so that overall the decoder appears dead to this test. Defining security relative to this notion of liveness would not guarantee any traceability for such decoders. Yet, such decoders would reasonably be considered useful, and would ideally be traced.

Motivated by the above discussion, we now give a "wish list" of features a liveness measurement should posses:

- It should collapse to the classical notion of goodness for a classical decoder.
- It should be "encoding independent". That is, if we apply some quantum transformation to the decoder's state (that gets undone when running the decoder), this should not affect the goodness of the decoder.
- If the same measurement is applied twice in a row (without any operations in between), it should return the same outcome both times. In other words, if a decoder is measured to be live, the resulting decoder should still be live.
- It should label decoders that start off with a high probability of decryption live, even if the decoder starts failing later.

Idea 2: Projective Implementations. In order to describe our solution, we recall some basic quantum measurement theory. A quantum state is simply a complex unit vector $|\psi\rangle$ of dimension d. For example, if the state consists of k qubits, d will be 2^k, with the components of $|\psi\rangle$ specifying weights for each of the d possible k-bit strings.

Any quantum measurement can be described as a positive operator valued measure (POVM). Such a measurement $\mathcal{M}$ is described by n Hermitian positive semi-definite matrices $M_1, \ldots, M_n$ such that $\sum_i M_i = \mathbf{I}$. When applying $\mathcal{M}$ to $|\psi\rangle$, the measurement results in outcome i with probability $p_i = \langle\psi|M_i|\psi\rangle$. The normalization on $|\psi\rangle$ and $\mathcal{M}$ ensures that this is a valid probability distribution. We stress that the matrices M_i and the weights in the vector $|\psi\rangle$ are not explicitly written out, but are implicitly defined by the measurement apparatus and the procedure that generates $|\psi\rangle$.

In our setting, we have the following POVM measurement: encrypt a random message bit m, run the pirate decoder on the resulting ciphertext, and then output 1 or 0 depending on whether the decoder correctly decrypts or not.

While the POVM formalism describes the probability distribution of the measurement, it does not describe the post-measurement quantum state. Indeed, many measurement apparatus could yield the same POVM, but result in different post-measurement states. A *general quantum measurement*, in contrast, determines both the measurement outcomes and the post-measurement states.

Our goal, given a POVM $\mathcal{M}$ and a state $|\psi\rangle$, is to learn the probability distribution from applying $\mathcal{M}$ to $|\psi\rangle$. The discussion above demonstrates that the actual probability distribution is information-theoretically hidden and inaccessible. Instead, we want a measurement $\mathcal{M}'$ that *measures* the distribution, such that $|\psi\rangle$ is a superposition over states with "well-defined" output distributions.

We interpret the above as the following. For a POVM $\mathcal{M}$ over outputs $\{1, \ldots, n\}$, we want a measurement $\mathcal{M}'$ which outputs a distribution (as in, it outputs a probability vector) over $\{1, \ldots, n\}$ such that $\mathcal{M}$ generates the same distribution of outputs as the following procedure:

- First, measure $\mathcal{M}'$ to obtain an observed distribution D
- Then sample a random value in $\{1, \ldots, n\}$ according to D.

Additionally, we want that subsequently applying $\mathcal{M}$ to the post-measurement state will yield exactly the distribution D, corresponding to measuring a decoder as live actually yielding a live decoder. We will call $\mathcal{M}'$ the *projective implementation* of $\mathcal{M}$ [3]. See Section 3 for a precise definition.

For general $\mathcal{M}$, there is no way to come up with a projective implementation $\mathcal{M}'$. In fact, we show that the existence of $\mathcal{M}'$ is equivalent to the matrices $M_1, \ldots, M_n$ in $\mathcal{M}$ all commuting, and when it exists $\mathcal{M}'$ is unique. Concretely, $\mathcal{M}'$ is the projective measurement in the simultaneous eigenbasis of the $M_1, \ldots, M_n$.

In our case, $\mathcal{M}$ has two outcomes, either correct or incorrect decryption, and normalization ($M_0 + M_1 = \mathbf{I}$) implies that M_0 and $M_1 = \mathbf{I} - M_0$ always commute. Therefore, $\mathcal{M}'$ must exist. Our test of liveness, essentially, will perform the measurement $\mathcal{M}'$ to get a distribution over $\{0, 1\}$—which is equivalent to measuring a success probability p—and then output "live" if p is sufficiently large; otherwise it outputs "dead".

We note that this liveness measurement satisfies all of our "wish list" items. In the case of classical decoders, M_0, M_1 are diagonal matrices whose entries are the success probabilities of the various classical decoders. As such, our projective implementation reduces to the classical goodness notion. Applying any encoding to the decoder state simply rotates the eigenbases of (M_0, M_1), but our notion automatically adjusts to such a rotation. The measurement is projective, implying that applying it twice will always yield the same answer. Finally, the notion captures the success probability of decrypting the very first ciphertext, and is not dependent on any subsequent decrypting abilities.

1.2.2 Our Quantum Tracing Model

With a notion of liveness in hand, we now turn to our tracing model. Even in the classical case there are potentially multiple tracing models. The most permissive for the tracing algorithm is to give the tracer the entire code of the decoder. This tracing model captures the setting where the decoder is an actual piece of software that the tracer has access to. Analogously, in the quantum setting we could give the tracer the actual quantum state representing the decoder, corresponding to a quantum software model.

On the other hand, over twenty-plus years of work on classical traitor tracing, the community has largely settled on a weaker "black box" model where the tracer can only query the decoder on ciphertexts and see the responses, but otherwise cannot observe how the decoder works. This is motivated in part due to the possibility of the decoder being *obfuscated* [BGI+01, GGH+13]—which informally hides everything about a program except for its input/output behavior—meaning the tracing algorithm does not gain much by inspecting the decoder's code. Moreover, in many cases it is desirable to trace an actually physical decoder box constructed by the traitors. In this case, various hardware security measures might be in place to prevent inspecting the decoder's operation.

In the black box setting, however, it is trivial to devise untraceable decoders: the decoder simply maintains a counter and ceases to function after a certain number of decryptions. If the number of ciphertexts the decoder decrypts is set small enough, tracing will become impossible. Such decoders are clearly less useful to pirates, but nonetheless represent a way for traitors to evade tracing.

[3] This terminology comes from the fact that we will ultimately set $\mathcal{M}'$ to be a "projective" measurement.

The classical solution is to restrict attention to *stateless* decoders. The implicit assumption in this model is that the tracer has some way to reset or rewind the decoder to its original state. In the software setting, such resets are trivial. Such resets may also be plausible—perhaps a hard reboot or cutting power will cause the counter to reset—depending on the hardware employed by the traitors.

Motivated by the years of development leading to the classical black box stateless decoder model, we would like to develop an analogous model for quantum decoders. However, we immediately face a definitional issue: for a general quantum decoder, it may be information-theoretically impossible to rewind the decoder to its initial state. This holds true even if we consider the software setting where the tracer has complete unfettered access to the decoder.

Our solution. We now describe our solution. Recall that, outside of measurements, quantum operations are indeed reversible. Therefore, we can imagine running the decoder until the measurement that produces the decrypted value. Then, we assume the ability to run all the operations, save for the final measurement, in reverse. This rewinding cannot possibly recover the initial state of the decoder, but in some sense it represents the closest we can get to a full rewinding. For example, in this model, the decoder's operation is "projective," which implies that a second decryption of the *same* ciphertext immediately following the first actually will *not* further alter the decoder's state, and moreover is guaranteed to give the same output. Analogous to the gentle measurement lemma [Aar04], if a particular decoder output occurs with overwhelming probability, a projective measurement will only negligibly affect the decoder's state. In particular, such projections collapse to the notion of stateless decoders in the classical setting.

Our black box decoder model therefore assumes that the decoder's operation is a projective measurement. Our precise formalization of a quantum black box model is somewhat delicate; see Section 4 for details. At a high level, what gets measured is not the adversary's output itself, but rather the single bit indicating whether the decoder was correct. This is done partly to accommodate relaxed decoder models from the classical literature [GKW18], and also motivated by the level of access that our ultimate tracing algorithm will need.

Our black box quantum decoder model is a natural generalization of the classical stateless decoder model. However, it remains to be seen whether it actually represents a realistic model of quantum hardware devices. Nevertheless, we emphasize that in the setting of quantum *software* decoders, it is always possible to perform the rewinding to implement a projective decoder. As a result, our model at least captures what is possible in the software setting.

1.2.3 Negative Result For Classical Black Box Tracing

One may hope that existing classical tracing algorithms for stateless decoders might also work for projective decoders, or at least that alternate classical tracing[4] could be devised. We show, unfortunately, that such classical tracing is unlikely. Concretely, for any $0 < \epsilon < 1/2$, we devise a quantum projective black box pirate decoder such that:

- The decoder starts out with decryption probability at least $1/2 + \epsilon$.
- For any polynomial-length sequence of classical ciphertext queries, there is a non-negligible probability that the decoder will respond to all queries with 0.

[4]By classical tracing, we mean that the tracer only queries the decoder on classical ciphertexts, and then uses the classical outputs in some way to accuse a user.

If the decoder outputs zero on all queries, it is clearly impossible to trace. The usual classical notions of tracing require that the tracing algorithm identifies a traitor with overwhelming probability by making $q = \mathsf{poly}(1/\epsilon)$ queries. Our counterexample would invalidate this definition.

We note that the definition of tracing could be relaxed to allow for some inverse polynomial probability τ that tracing fails, and then allow the number of queries by the tracer to be $q = \mathsf{poly}(1/\epsilon, 1/\tau)$. Our counterexample does not rule out such a weaker tracing notion. Nevertheless, our counter example shows that the existing guarantees of classical tracing algorithms do not carry over to the quantum projective decoder setting. Additionally, it shows that if one wants to achieve the strong tracing guarantees analogous to tracing classical decoders, the tracing algorithm should make *quantum* queries to the decoder. Thus, our model of black box decoders will allow for such quantum queries. Again, such queries are always possible for software decoders.

1.2.4 Our Quantum Tracing Algorithm

We now turn to our tracing algorithm. We observe that essentially all classical traitor tracing solutions work abstractly as follows: the tracer generates ciphertexts from invalid distributions D_S for various subsets S of users, where decryption is possible only for users in S. An additional guarantee is typically that only users in the symmetric difference of S and T can distinguish D_S from D_T. The tracer estimates the probabilities $\hat{p}_S$ that the pirate decoder decrypts D_S by testing the decoder on various samples from D_S. Typically, the first S is the set of all users, corresponding to D_S being all valid ciphertexts. Subsequently, additional sets S are considered. Large gaps between the $\hat{p}_S$ then give information about the identities of the traitor(s).

This framework is very broad, encompassing essentially the entire body of traitor tracing literature. For example, it encompasses the private linear broadcast encryption (PLBE) approach of [BSW06], which is the backbone of most of the various algebraic traitor tracing constructions [BSW06, GGH+13, BZ14, GKW18]. Here, the sets S have the form $[i] = \{1, 2, \ldots, i\}$ for various i. This framework also encompasses combinatorial schemes such as [CFN94, BN08]. For example, the most basic scheme of [CFN94] uses the bit-fixing sets $S_{i,b} = \{x \in \{0,1\}^k : x_i = b\}$. The fingerprinting code-based construction of [BN08] uses a set structure that is actually kept secret, except to the tracer.

Our goal will be to upgrade classical tracing algorithms to work with quantum decoders. As we will see, there are numerous problems that must be overcome.

Approximating $\mathcal{M}'$ efficiently. We first aim to build a quantum analog of this classical probability estimation. For exactly the same reasons encountered when defining traitor tracing, the actual success probabilities $\hat{p}_S$ cannot be accessed in any physical way for a quantum decoder. As in the discussion leading to our tracing definition, the most natural alternative is to instead *measure* the success probability, obtaining a measurement p_S. In the case of S being all users, this means the tracing algorithm would need to implement the measurement $\mathcal{M}'$ from above, and for other S analogous measurements will be needed.

However, while a projective implementation $\mathcal{M}'$ is guaranteed to exist, we have not guaranteed that it is computationally efficient. In fact, it *cannot* be computationally efficient, even classically. This is simply because, even classically, we cannot efficiently learn the exact output distribution of a program[5]. Classically, this is resolved by having the tracer *estimate* the success probability of the

[5]This means that the security experiment is inefficient. However, the same is true of classical traitor tracing

decoder, and demonstrating that an estimate is good enough for tracing.

We would therefore like to develop a procedure that approximates the measurement $\mathcal{M}'$. Yet the matrices M_i are exponentially-large, being only implicitly defined by the measurement apparatus of the decoder. Therefore, eigen-decomposition would be intractable. Our negative result also means cannot use classical estimation techniques, since those work by running the decoder on classical ciphertexts.

Instead, we devise an operation on the quantum pirate decoder that tries ciphertexts *in superposition*; our operation will still work in the black box projection model for pirate decoders, which allows for such quantum queries. Our algorithm makes use of the fact that $\mathcal{M}'$ is *projective*. More precisely, if $\mathcal{M}_c = (M_{c,0}, M_{c,1})$ is the measurement which tests if the decoder correctly decrypts c, then $\mathcal{M}_c$ is guaranteed to be projective by our decoder model. The overall measurement POVM $\mathcal{M}_c = (M_0, M_1)$ for testing correctness on a random ciphertext is then the *average* or *mixture* of the $\mathcal{M}_c$:

$$M_b = \sum_c \Pr[c] M_{c,b} \ .$$

Our black box decoder model allows us to evaluate the projective $\mathcal{M}_c$ for any ciphertext c, or even evaluate the $\mathcal{M}_c$ for *superpositions* of c values. We demonstrate how to use this ability to compute an approximation of $\mathcal{M}'$.

To do so, we employ a technique of Watrous and Marriott [MW04], which was originally used for decreasing error in quantum Arthur-Merlin games. We show that their algorithm, with some small modifications, works in our setting to achieve a reasonable approximation of $\mathcal{M}'$. At a very high level, the algorithm runs $\mathcal{M}_c$ over a superposition of c, and getting a measurement outcome b_1. Then we apply a particular measurement to the superposition of c, obtaining measurement d_1. We interleave and repeat both measurements a number of times, obtaining a sequence $d_0 = 0, b_1, d_1, b_2, d_2 \ldots$. The output is p' where $1 - p'$ is the fraction of bit flips in the sequence.

Following the analysis from [MW04], we show that the output of this measurement indeed approximates the distribution of $\mathcal{M}'$. One wrinkle is that [MW04] did not care about the post-measurement state of the decoder, whereas we want the post-measurement states for $\mathcal{M}'$ and the approximation to be "close" in some sense. We show that, by being careful about exactly when the sequence of measurements is terminated, we can guarantee the necessary closeness.

On Computational Indistinguishability. Recall that, in addition to estimating probabilities p_S, classical tracing algorithms typically rely on p_S and p_T being close for different sets S, T, as long as the adversary controls no users in the symmetric difference between S, T. Classically, such closeness follows readily from the indistinguishability between (many samples of) D_S, D_T. Indeed, if p_S, p_T were far, a distinguisher could use the samples to compute an estimate of the success probability, and then guess which distribution the samples came from.

Quantumly, such closeness is non-obvious. Since the POVMs corresponding to D_S, D_T simply run the decoder on a single classical ciphertext, we know that the probability the decoder is correct on the two distributions must be close. This implies that the *means* of the distributions on p_S and p_T must be close. But this alone is insufficient. For example, for a given decoder, p_S might be always measure to be $3/4$, whereas p_T measures to be $1/2$ or 1 with equal probability. Both distributions have the same mean, but are nevertheless far apart.

experiments for essentially the same reason.

Now, our algorithm for approximating the projective implementation allows us to efficiently estimate p_S or p_T, which would therefore allow us to distinguish the two cases above. However, our algorithm runs the decoder on *quantum superpositions* of exponentially-many ciphertexts, and this quantumness is somewhat inherent, per our negative result. But perhaps such superpositions are actually distinguishable, even if the individual ciphertext samples are not? For example, [GKZ19] shows that superpositions over LWE samples can be distinguished, despite individual samples being presumably indistinguishable.

We show that, nonetheless, if polynomially-many samples of D_S and D_T are computationally indistinguishable, then the distributions over measured p_S and p_T must be close, in some sense [6]. We show this by a careful application of the small-range distributions of Zhandry [Zha12a]. These distributions allow us to approximate the measurements of p_S or p_T using only a polynomial number of classical samples from either ciphertext distribution.

Handling Non-simultaneous Measurements. Based on the above indistinguishability result, we know, for a given decoder state, that p_S and p_T being far means the attacker must in fact control a user in the symmetric difference between S and T. As in the classical case, we would therefore like to use this information to narrow down our list of suspected traitors. Unfortunately, we cannot actually simultaneously measure p_S and p_T for the same state: once we measure one of them, say p_S, the decoder state is potentially irreversibly altered. If we then measure p_T, we will get a result, but p_T and p_S will be measurements from different states, and it is not obvious what comparing p_S and p_T yields.

Nevertheless, we show that if p_S and p_T are measured in succession, and if the underlying distributions D_S and D_T are indistinguishable (for polynomially many samples), then p_S and p_T will in fact be close. Supposing we applied the actual projective implementation corresponding to D_S, we know that the resulting decoder $|☠_S\rangle$ is an eigenstate of the measurement. Thus, if we applied the projective implementation a second time to $|☠_S\rangle$, obtaining a second measurement p'_S of $\hat{p}_S$, then $p_S = p'_S$. We show that if we relax to using our approximation algorithm, then $p'_S \approx p_S$. If we replace this second measurement on $|☠_S\rangle$ with our approximation of p_T, then by our computational indistinguishability guarantee, $p_T \approx p'_S \approx p_S$ (notice that p'_S is never actually computed; it is just used in the analysis). Thus, if p_S and p_T are far, the adversary must control a user in the symmetric difference between S and T, as desired.

How to Trace PLBE. Up until this point, our discussion has applied broadly to most tracing algorithms and one may hope to simply swap out the probability estimation steps of classical tracing algorithms with our approximate projective implementation algorithm. Unfortunately, this does not appear to work in general. To see the issue, consider a tracing algorithm which first computes (an estimate of) p_S. We know that the decoder is live, so $p_{[N]}$ (the success probability for valid ciphertexts) must be noticeably higher than $1/2$; let's say $p_{[N]} = 1$. Suppose p_S is measured to be $1/2 \ll p_{[N]}$. We therefore know that the adversary must control a user in $[N] \setminus S$. However, this might not be sufficient for accusing a user: perhaps S only contains $N/2$ users, in which case we have only narrowed the attacker down to half the users. Tracing must then proceed to compute p_T for a different set T. But at this point, perhaps the decoder has actually collapsed to a dead decoder and we can no longer learn any information from it.

[6] Statistical closeness is too-strong a requirement, which is also true classically. Instead, we consider a weaker notion of distance based on the Euclidean distance

The takeaway is: the very first time any gap is found, the decoder could potentially now be dead, and we should therefore be ready to accuse a user. In the example above, if S contained all but one user, say user N, we could then immediately accuse user N. We would then satisfy the desired tracing guarantee, despite having a now-useless decoder. If on the other hand p_S were measured to be greater than $1/2$, we can continue to measure p_T. The same issue occurs if there is more than one user in $S \setminus T$, so we would want to have T contain all users in S except a single user, say user $N-1$.

What is needed, therefore, is a linear set structure, where it is possible to encrypt to subsets $[j]$ of users, $j = N, N-1, \ldots, 0$, where users $i \leq j$ can decrypt, users $i > j$ cannot, and it is impossible to distinguish $[j]$ from $[j-1]$ unless the adversary controls user j. In other words, we need *private linear broadcast encryption* (PLBE) as defined by [BSW06]. Based on the above, we show that any PLBE with the right properties (elaborated below) can be traced. Our tracing algorithm proceeds essentially as the classical tracing algorithm given in [BSW06], except that we use our quantum approximation algorithm to compute the various probabilities $p_{[j]}$. We also must compute the $p_{[j]}$ in a particular order, namely in order of *decreasing* j, whereas the order does not matter in [BSW06].

Applications and Limitations. Fortunately, PLBE is the most common approach to building traitor tracing, and therefore our tracing algorithm is broadly applicable. For example, sufficiently strong PLBE can be instantiated from

- Generic public key encryption, resulting in ciphertexts and public keys that grow linearly with the number of users.
- From post-quantum obfuscation [BGMZ18], following [GGH+13, BZ14], resulting in constant-sized ciphertexts.
- In the setting of bounded collusions, we can use bounded-collusion secure functional encryption, which can be instantiated from generic public key encryption [AV19]. The resulting scheme has ciphertexts growing linearly in the collusion bound (but independent of the total number of users).

We note that PLBE can also be constructed from pairings [BSW06], though this instantiation is not useful in our context since pairings are insecure against quantum attackers.

Unfortunately, our analysis does not seem to extend to a variant of PLBE that was recently constructed from LWE by Goyal, Koppula, and Waters [GKW18] for subtle reasons. Indeed, their version of PLBE has encryptions to sets $[j]$ for $j < N$ requiring a *secret encryption key*, and indistinguishability of $D_{[j]}$ and $D_{[j-1]}$ only holds for those who do not know the secret encryption key. The implication is that tracing can only be carried out by the holder of the secret key. The fact that tracing requires a secret key is itself not a problem for us, as we can similarly consider a secret key version of tracing. The issue is that, when we prove $p_{[j]}$ is close to $p_{[j-1]}$, we need indistinguishability between $[j]$ and $[j-1]$ to hold for polynomially ciphertexts. On the other hand, [GKW18] only remains secure for a constant number of ciphertexts, and the natural ways of extending [GKW18] to handle more ciphertexts will blow up the ciphertext too much. We therefore leave tracing quantum decoders for [GKW18] as an important open problem.

We also note that our approach does not appear to extend to combinatorial traitor tracing schemes, such as [CFN94, BN08]. In these schemes, the sets S do not have the needed linear structure. As discussed above, this means that the decoder could fail on the first distribution D_S for

$S \neq [N]$, and no longer work for *any* other distribution. Since $[N] \setminus S$ contains more than 1 identity, there is no way to accuse a user using our approach. We leave as an interesting open question developing a tracing algorithm for these combinatorial constructions, or alternatively demonstrating a quantum pirate decoder that cannot be traced.

1.3 Paper Outline

Section 2 gives a basic background in quantum notation and operations. In Section 3, we develop our notion of projective implementations, which will be used in Section 4 to define traitor tracing for pirate decoders. In Section 5, we demonstrate that quantum access to a quantum decoder is necessary for tracing. In Section 6, we develop our algorithm for estimating the success probability of a pirate decoder, which is then used in our tracing algorithm in Section 7.

2 Quantum Preliminaries

In this work, we will make use of two formalisms for quantum measurements. The first, a *positive operator valued measure* (POVM), is a general form of quantum measurement. A POVM $\mathcal{M}$ is specified by a finite index set $\mathcal{I}$ and a set $\{M_i\}_{i\in\mathcal{I}}$ of hermitian positive semidefinite matrices M_i with the normalization requirement $\sum_{i\in\mathcal{I}} M_i = \mathbf{I}$. The matrices M_i are called *items* of the POVM. When applying a POVM $\mathcal{M}$ to a quantum state $|\psi\rangle$, the result of the measurement is i with probability $p_i = \langle\psi|M_i|\psi\rangle$. The normalization requirements for $\mathcal{M}$ and $|\psi\rangle$ imply that $\sum_i p_i = 1$, and therefore this is indeed a probability distribution. We denote by $\mathcal{M}(|\psi\rangle)$ the distribution obtained by applying $\mathcal{M}$ to $|\psi\rangle$.

The POVM formalism describes the probabilities of various outcomes, but it does not specify how $|\psi\rangle$ is affected by measurement. Indeed, there will be many possible implementations of a measurement giving rise to the same probability distribution of outcomes, but resulting in different post-measurement states.

To account for this, the second formalism we will use is simply called a *quantum measurement*. Here, a quantum measurement $\mathcal{E}$ is specified by a finite index set $\mathcal{I}$ and a set $\{E_i\}_{i\in\mathcal{I}}$ of matrices E_i (not necessarily hermitian nor positive) such that $\sum_{i\in\mathcal{I}} E_i^\dagger E_i = \mathbf{I}$. The matrices E_i are called *measurement operators*. When applying a quantum measurement $\mathcal{E}$ to a quantum state $|\psi\rangle$, the result of the measurement is i with probability $p_i = \langle\psi|E_i^\dagger E_i|\psi\rangle = \||E_i|\psi\rangle\|^2$. Conditioned on the outcome being i, the post-measurement state is $E_i|\psi\rangle/\sqrt{p_i}$, where the factor $\sqrt{p_i}$ is to ensure that the state is normalized.

We note that any quantum measurement $\mathcal{E}$ is associated with a POVM $\mathcal{M} = \mathsf{POVM}(\mathcal{E})$ with $M_i = E_i^\dagger E_i$. We will call $\mathcal{E}$ an *implementation* of $\mathcal{M}$. We note that while each quantum measurement implements exactly one POVM, each POVM may be implemented by many possible quantum measurements.

A *projective measurement* is a quantum measurement where the E_i are projections: E_i are hermitian and satisfy $E_i^2 = E_i$. We note that $\sum_i E_i = \sum_i E_i^\dagger E_i = \mathbf{I}$ implies that $E_i E_j = 0$ for $i \neq j$.

A *projective POVM* is a POVM where M_i are projections. We note that the POVM associated with a projective measurement is projective. However, a projective POVM may be implemented by non-projective measurements. As with quantum measurements, a projective POVM will satisfy $M_i M_j = 0$ for $i \neq j$.

3 Commutative POVMs and Projective Implementations

In this section, we give some additional definitions for quantum measurements and POVMs, as well as some basic results. In Section 4, we use these definitions and results to define our notion of traitor tracing for pirate decoders.

Definition 3.1. *A POVM $\mathcal{M} = \{M_i\}_{i\in\mathcal{I}}$ is* commutative *if $M_iM_j = M_jM_i \forall i, j$.*

Let $\mathcal{I}$ be an index set, and let $\mathcal{D}$ be a finite set of distributions over $\mathcal{I}$. Let $\mathcal{E} = \{E_D\}_{D\in\mathcal{D}}$ be a projective measurement with index set $\mathcal{D}$. Consider the POVM $\mathcal{M} = \{M_i\}_{i\in\mathcal{I}}$ where $M_i = \sum_{D\in\mathcal{D}} E_D \Pr[D = i]$. Then $\mathcal{M}$ is equivalent to the following measurement process:

- First apply the measurement $\mathcal{E}$ to obtain a distribution D
- Then choose a random sample i according to D

Definition 3.2. *For $\mathcal{E}, \mathcal{M}$ be as above, $\mathcal{E}$ is the* projective implementation *of $\mathcal{M}$.*

Lemma 3.3. *A POVM $\mathcal{M} = \{M_i\}_{i\in\mathcal{I}}$ is commutative if and only if it has a projective implementation; the projective implementation is unique.*

Proof. First, note that for any projective measurement $\mathcal{E}$, the measurement operators must commute. Therefore, if $\mathcal{M}$ has a projective implementation $\mathcal{E}$, we have $M_iM_j = \sum_{D_1,D_2} E_{D_1}E_{D_2} \Pr[D_1 = i] \Pr[D_2 = i] = \sum_{D_1,D_2} E_{D_2}E_{D_1} \Pr[D_1 = i] \Pr[D_2 = i] = M_jM_i$, meaning $\mathcal{M}$ is commutative.

Next, suppose $\mathcal{M}$ is commutative. Then the M_i are simultaneously diagonalizable, meaning there is an orthonormal basis $|b_1\rangle, \ldots, |b_m\rangle$ for the Hilbert space such that each $|b_j\rangle$ is an eigenvector of each M_i. For each $|b_j\rangle$, let D_j be the distribution of outputs of $\mathcal{M}$ when applied to state $|b_j\rangle$. Then notice that $|b_j\rangle$ is an eigenvector of M_i with eigenvalue $\Pr[D_j = i]$.

Next, let $\mathcal{D} = \{D_j\}$ be the set of distributions D_j. Note that there may be multiple j giving the same D_j; we only include such distributions *once* in $\mathcal{D}$ so that $\mathcal{D}$ remains a set. Let $E_D = \sum_{j:D_j=D} |b_j\rangle\langle b_j|$. Then let $\mathcal{E} = \{E_D\}_{D\in\mathcal{D}}$. Then notice that $\sum_D E_D \Pr[D = i] = \sum_j |b_j\rangle\langle b_j| \sum_i \Pr[D_j = i] = M_i$, so $\mathcal{E}$ is a projective implementation of $\mathcal{M}$.

For uniqueness, consider a projective measurement $\mathcal{E}$. If we construct $\mathcal{M}$ as above, and then run the procedure to get a projective implementation $\mathcal{E}'$, it is straightforward that $\mathcal{E}' = \mathcal{E}$. □

Therefore, for a commutative POVM $\mathcal{M}$, we will let $\mathsf{ProjImp}(\mathcal{M})$ denote the unique projective measurement.

4 Defining Tracing of Quantum Pirates

4.1 Traitor Tracing Syntax

Here, we give the syntax for public key traitor tracing with public traceability. Variants with secret key encryption and/or secret key tracing are defined analogously. A traitor tracing system is a tuple of four algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ defined as follows:

- $\mathsf{Gen}(1^\lambda, 1^N)$ is a classical probabilistic polynomial time (PPT) algorithm that takes as input the security parameter and a number N of users, and samples a public key pk, and N secret keys $\mathsf{sk}_1, \ldots, \mathsf{sk}_N$.

- $\mathsf{Enc}(\mathsf{pk}, m)$ is a classical PPT algorithm that takes as input the public key pk and a message m, and outputs a ciphertext c.
- $\mathsf{Dec}(\mathsf{sk}_i, c)$ is a classical deterministic algorithm that takes as input a secret key sk_i for user i and a ciphertext, and outputs a message m'.
- $\mathsf{Trace}^{|\text{☠}\rangle}(\mathsf{pk}, m_0, m_1, \epsilon)$ takes as input the public key pk, two messages m_0, m_1, and a parameter ϵ. It makes queries to a pirate decoder $|\text{☠}\rangle$. It ultimately outputs a subset of $[N]$, which are the accused users.

4.2 Decoder Models

We now specify $|\text{☠}\rangle$ and what a query to $|\text{☠}\rangle$ does. $|\text{☠}\rangle$ consists of a collection of qubits $|\psi\rangle$ and the description of an efficient procedure U. U maps a ciphertext c to an efficiently computable unitary operation $U(c)$ which acts on $|\psi\rangle$.

The assumed operation of the decoder in this model, denoted $\mathsf{Eval}^{|\text{☠}\rangle}(c)$, is the following: on input a ciphertext c, compute $U(c)$. Then apply $U(c)$ to $|\psi\rangle$. Finally, measure the first qubit of $U(c)|\psi\rangle$, and output the result.

In the classical setting, various levels of access to the decoder may be possible. For example, the decoder may be a digital program, and the tracer actually obtains the program code. Alternatively, the decoder may be an actually physical piece of hardware, and the tracer has only access to the input/output behavior. In the quantum setting, one can imagine analogous scenarios. Below, we describe decoder models to capture some scenarios in the quantum decoder setting.

Software Decoder Model. The Software Decoder model will be the quantum analog of the classical setting where the decoder is a software program. In this model, a query to $|\text{☠}\rangle$ consists of the empty string ϵ, and in response the Trace receives the entire state $|\text{☠}\rangle$ (including U). In this sense, Trace has complete access to the entire decoder. Next, we will consider decoder models where Trace has limited access. Such models will be potential useful in hardware settings.

We now develop black box models of quantum decoders, which hopefully generalizes the classical notion of stateless decoders. Of course, some limitations of the decoder are necessary, to prevent simple counterexamples like self-destructing after a counter reaches a certain value. Our goal is to identify the minimal type of query access needed to allow tracing. We now discuss two different models.

The Black Box Unitary Model. Inspired by this rewinding view of black box stateless decoders, we consider a model where Trace has only black box access to the functionality U, as well as $U^\dagger$ for rewinding. Additionally, Trace does not have direct access to $|\psi\rangle$, and can only operate on $|\psi\rangle$ through $U, U^\dagger$. More precisely, Trace can make three types of queries to $|\text{☠}\rangle$:

- Output queries. Here, Trace prepares and sends a qubit. A controlled NOT (CNOT) is applied to the qubit, with the control bit is the first qubit of $|\psi\rangle$.
- U queries. Here, Trace prepares a superposition $\sum_{\mathsf{aux},c} \alpha_{\mathsf{aux},c}|\mathsf{aux}, c\rangle$, where c ranges over ciphertexts and aux corresponds to the private state that Trace keeps to itself. $|\text{☠}\rangle$ performs

the following operation:

$$\sum_{\mathsf{aux},c} \alpha_{\mathsf{aux},c}|\mathsf{aux},c\rangle \otimes |\psi\rangle \mapsto \sum_{\mathsf{aux},c} \alpha_{\mathsf{aux},c}|\mathsf{aux},c\rangle \otimes U(c)|\psi\rangle$$

- $U^\dagger$ queries. This is the same as a U query, except that the inverse operation $U^\dagger$ is applied: U queries. Here, Trace prepares a superposition $\sum_{\mathsf{aux},c} \alpha_{\mathsf{aux},c}|\mathsf{aux},c\rangle$, where c ranges over ciphertexts and aux corresponds to the private state that Trace keeps to itself. $|☠\rangle$ performs the following operation:

$$\sum_{\mathsf{aux},c} \alpha_{\mathsf{aux},c}|\mathsf{aux},c\rangle \otimes |\psi\rangle \mapsto \sum_{\mathsf{aux},c} \alpha_{\mathsf{aux},c}|\mathsf{aux},c\rangle \otimes U^\dagger(c)|\psi\rangle$$

The following lemma is immediate:

Lemma 4.1. *Let $A^{|☠\rangle}(\cdot)$ be any quantum polynomial-time algorithm that takes as input x and makes queries to $|☠\rangle$ in the Black Box Unitary model. Then there exists another quantum polynomial-time algorithm $B^{|☠\rangle}(\cdot)$ in the Software Decoder model such that, for any x, y,*

$$\Pr[A^{|☠\rangle}(x) = y] = \Pr[B^{|☠\rangle}(x) = y]$$

In other words, a tracing scheme in the Black Box Unitary model can also trace in the Software Decoder model.

The Black Box Projection Model. The Black Box Unitary model may still not be possible for a given decoder device, as it still requires having access to U and $U^\dagger$, as opposed to the input/output functionality of the device. Here, we seek to provide a minimal type of query access that more closely resembles the input/output functionality, and is therefore more likely to be physically realizable. Of course, *some* assumption on the internal workings of the decoder are necessary, to prevent decoders that simply self-destruct after a single query by toggling some control bit.

In our "black box projection model", a query to $|☠\rangle$ has the form $\sum_{\mathsf{aux},c,b} \alpha_{\mathsf{aux},c,b}|\mathsf{aux},c,b\rangle$, where c ranges over ciphertexts, b over bits, and aux over an arbitrary domain. In response to the query, $|☠\rangle$ does the following:

1. First, it performs the following action on basis states:

$$|\mathsf{aux},c,b\rangle \otimes |\psi\rangle \mapsto |\mathsf{aux},c,b\rangle \otimes U(c)|\psi\rangle \ .$$

2. Apply a controlled NOT (CNOT) to the b register, where the control bit is the first qubit of the decoder's state.

3. Next, it applies the inverse of Step 1:

$$|\mathsf{aux},c,b\rangle \otimes |\psi\rangle \mapsto |\mathsf{aux},c,b\rangle \otimes U^\dagger(c)|\psi\rangle \ .$$

4. Finally, it measures the b register, and then returns the result b as well as whatever remains in the aux, c registers.

Note that the query is a projective measurement on $|\psi\rangle$. Recall that applying a projective measurement twice in a row will always result in identical outcomes. This is similar to how a classical stateless (deterministic) decoder will always produce the same outcome on repeated ciphertexts. Thus projective measurements are a generalization of stateless decoders, though other generalizations are possible.

Lemma 4.2. *Let $A^{|☠\rangle}(\cdot)$ be any quantum polynomial-time algorithm that takes as input x and makes queries to $|☠\rangle$ in the Black Box Projection model. Then there exists another quantum polynomial-time algorithm $B^{|☠\rangle}(\cdot)$ in the Black Box Unitary model such that, for any x, y, $\Pr[A^{|☠\rangle}(x) = y] = \Pr[B^{|☠\rangle}(x) = y]$.*

Since the Black Box Projection model is the weakest model we consider, ability to trace in this model gives the strongest guarantees. We now discuss some of the choice made in our Black Box Projection model. Therefore, for the rest of this paper, we will focus on the Black Box Projection model.

Superposition Queries. Our model allows queries on superpositions of ciphertexts. We could have instead required classical queries. Unfortunately, such a model seems untraceable, evidenced by our negative result in Section 5.

Returning the ciphertext registers. One could alternatively only return b' and not the ciphertext (the aux registers being held privately by Trace). This is equivalent to measuring the ciphertext, resulting in effectively a classical query model.

The role of b. An alternative is to measure the first qubit of the decoder's state directly (that is, the intended output of the decoder), instead of measuring the result of XORing with b. We have two reasons for our modeling choice:

- The standard query model for quantum operations has the query response XORed into some registers provided as part of the query. Our modeling mimics this query behavior. We thus have the measurement applied to only the output of the decoder in the XOR query model, rather than having the measurement applied to the private state of the decoder.
- If we initialize the b registers to initially contain the correct answer expected from the decoder, the result of the query measurement will tell us whether the decoder answered correctly or incorrectly, as opposed to telling us the actual answer. This turns out to be crucial for our tracing algorithm. Indeed, as we will see in Section 6, the given Black Box Projection model will allow us to measure the success probability of the decoder. On the other hand, if the measurement were applied directly to the decoder state, we would be able to measure either of the probabilities p_r that the decoder outputs 1 on a random encryption of the bit r. To to get the success probability, we would need to know both p_0 and p_1. But in the quantum case it may not be possible to learn both values simultaneously if the measurements are "incompatible."

4.3 Correctness and Security.

We have the usual correctness requirement:

Definition 4.3. *A traitor tracing system is* correct *if, for all messages m and functions $N = N(\lambda), i = i(\lambda)$,*

$$\Pr[\mathsf{Dec}(\mathsf{sk}_i, \mathsf{Enc}(\mathsf{pk}, m)) = m : (\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}(\lambda, N)] \geq 1 - \mathsf{negl}(\lambda)$$

For brevity, we omit the semantic security requirement and focus on tracing. Our definition is inspired by that of [GKW18], adapted to use our decoder model. For a decoder $|☠\rangle = (U, |\psi\rangle)$, two messages m_0, m_1, consider the operation on $|\psi\rangle$:

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

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

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

- A gets λ, and replies with a number N. Both λ, N are represented in unary.
- Run $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$, and send pk to A.
- A then makes an arbitrary number of classical queries on identities $i \in [N]$; in response it receives sk_i. Let S be the set of i queried by A.
- Next, A outputs $(|☠\rangle, m_0, m_1)$ for decoder $|☠\rangle$ and messages m_0, m_1.

Now consider two possible operations on $|☠\rangle$:

- $S' \leftarrow \mathsf{Trace}^{|☠\rangle}(\mathsf{pk}, m_0, m_1, \epsilon)$. Let $\mathsf{BadTrace}$ as the event that $S \setminus S' \neq \emptyset$. We define the event $\mathsf{GoodTrace}$ as the event that $S' \neq \emptyset$
- Apply the measurement $\mathcal{M}'$ to $|☠\rangle$, obtaining a probability p. Let Live be the event that $p \geq 1/2 + \epsilon$.

Definition 4.4. *A tracing system is* quantum traceable *if for all quantum polynomial time adversaries A and for every inverse polynomial ϵ, there is a negligible* negl *such that* $\Pr[\mathsf{BadTrace}] < \mathsf{negl}(\lambda)$ *and* $\Pr[\mathsf{GoodTrace}] \geq \Pr[\mathsf{Live}] - \mathsf{negl}(\lambda)$.

Remark 4.5. *We note that in general, $\mathcal{M}'$ is inefficient, as it involves computing eigenvalues of an exponentially large implicitly represented matrix. Such inefficiency, however, is typical of classical traitor tracing definitions. In particular, classical tracing definitions have the challenger compute the success probability of the decoder, which is similarly an inefficient process. The measurement $\mathcal{M}'$ is our quantum analog of such classical success probability calculations, and is therefore unsurprisingly*

inefficient as well. In classical security proofs, the success probability calculation is replaced with an efficient estimation, thereby making the reduction efficient despite the inefficient definition. Similarly, in Section 6, we will demonstrate how to efficiently approximate $\mathcal{M}'$ in order to achieve an efficient reduction.

Remark 4.6. *We note that the two operations $\mathcal{M}'$ and* Trace *cannot both be simultaneously applied to the decoder. Therefore, we cannot consider the events* BadTrace, GoodTrace *and* Live *simultaneously. However, we can still reason about the probabilities of such events happening, conditioned the appropriate operation being performed. The condition* $\Pr[\mathsf{GoodTrace}] \geq \Pr[\mathsf{Live}] - \mathsf{negl}(\lambda)$ *says that tracing will accuse a user essentially as often as the decoder would be deemed live. This probability-comparison way of defining traitor tracing is in the spirit of recent definitions from [GKRW18, GKW18].*

We note that classically, another way to define security is to require that if Live *happens, then* GoodTrace *must happen also. Since the adversary can estimate* Live *efficiently for himself, these this alternative definition is essentially equivalent to the definitions from [GKRW18, GKW18]. Similarly in the quantum setting, we could have defined traitor tracing to first apply $\mathcal{M}'$, and then trace, and ask that a user is accused whenever $\mathcal{M}'$ outputs a high-enough probability. This definition would similarly be essentially equivalent to our given definition, since the adversary can estimate* Live *efficiently for himself using our procedure in Section 6. Our given definition, however, will make our much proofs cleaner.*

5 On the Necessity of Quantum Queries

Here, we address the need for quantum queries in tracing. We consider a variant of our Black Box Projection model where queries to the decoder are only on *classical* ciphertexts c. Concretely, when a query is made to $|☠\rangle$, the c registers are additionally measured, to ensure that only a classical ciphertext is input. We call this the Classical Black Box Projection model.

Theorem 5.1. *Any traitor tracing scheme which operates in the Classical Black Box Projection model is* not *quantum traceable according to Definition 4.4.*

Proof. We will assume for simplicity that the traitor tracing scheme has perfect correctness: $\mathsf{Dec}(\mathsf{sk}_j, \mathsf{Enc}(\mathsf{pk}, m)) = m$ with probability 1. We will also assume for any m, and any two ciphertexts c, c' that are both valid encryptions of m, that $\Pr[c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)] = \Pr[c' \leftarrow \mathsf{Enc}(\mathsf{pk}, m)]$. In other words, the distribution on ciphertexts is uniform over the support of valid encryptions of m. It is straightforward but slightly tedious to generalize to arbitrary tracing schemes.

First, suppose there exists a message m such the guessing probability of encryptions of m is non-negligible. That is, $\Pr[c = \mathsf{Enc}(\mathsf{pk}, m)]$ is non-negligible for some c. Then the scheme cannot have semantic security. In this case, even classically tracing is impossible, since an adversary with no keys at all can build a useful pirate decoder that simply rejects all ciphertexts but c, and on c outputs m.

From now on, we will assume that for all messages, the ciphertexts have negligible guessing probability. Fix an inverse polynomial ϵ that is bounded below $1/2$, and polynomial N. We construct an adversary A which chooses a random $j \in [N]$, and queries for secret key sk_j. It then chooses two arbitrary distinct messages m_0, m_1 and constructs the following decoder $|☠\rangle$. First let

$$\mathsf{Dec}'(c) := \begin{cases} b & \text{if } \mathsf{Dec}(\mathsf{sk}_j, c) = m_b \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathcal{H}$ have basis $\{|c\rangle\}_c \cup \{|\perp\rangle\}$, where c ranges over all possible ciphertexts. The decoder's initial state is $|\perp\rangle \otimes |0\rangle |m_0, m_1, \mathsf{sk}_j\rangle$. That is, the decoder's state consists of the system $\mathcal{H}$ initialized to $|\perp\rangle$, a qubit $\mathcal{H}_2$ initialized to $|0\rangle$, as well as the messages m_0, m_1 and the secret key sk_j. Define the vectors $|\phi_c\rangle \in \mathcal{H}$ as $|\phi_c\rangle = \sqrt{2\epsilon}|\perp\rangle + \sqrt{1-2\epsilon}|c\rangle$. Let $U(c)$ be the unitary over $\mathcal{H}_2 \otimes \mathcal{H}$:

$$U(c) = (|1 - \mathsf{Dec}'(c)\rangle\langle 1| + |\mathsf{Dec}'(c)\rangle\langle 0|) \otimes |\phi_c\rangle\langle\phi_c| + \mathbf{I} \otimes (\mathbf{I} - |\phi_c\rangle\langle\phi_c|)$$

The output register for $|☠\rangle$ is set to $\mathcal{H}_2$. Informally, $U(c)$ applies the projective measurement $(P_c, Q_c = \mathbf{I} - P_c)$, where $P_c := |\phi_c\rangle\langle\phi_c|$. Then conditioned on the measurement output being 1, it XORs $\mathsf{Dec}'(c)$ into the output register.

Claim 5.2. *There exists a negligible function* negl *such that* $\Pr[\mathsf{Live}] \geq 1 - \mathsf{negl}$.

Proof. Let $\mathcal{C}_0$ be the valid encryptions of m_0, and $\mathcal{C}_1$ be the valid encryptions of m_1. Write $c \leftarrow \mathcal{C}_b$ as the distribution on c obtained by encrypting m_b. By perfect correctness, $\mathcal{C}_0 \cap \mathcal{C}_1 = \emptyset$. For $c \in \mathcal{C}_0$, carrying out the decoder query on c will have no effect on the state of the decoder and the decoder will always output 0, which is the correct answer. For $c \in \mathcal{C}_1$, then the decoder will output 1 (the correct answer) if and only if the measurement $(P_c, Q_c = \mathbf{I} - P_c)$ outputs 0.

Therefore, the associated POVM to the decoder then has the form $\mathcal{P} = (P = \mathbf{I} - Q, Q)$ where

$$Q = \frac{1}{2}\left(\mathbf{I} + \frac{1}{|\mathcal{C}_1|}\sum_{c\in\mathcal{C}_1} |\phi_c\rangle\langle\phi_c|\right)$$

Consider the vector

$$|s\rangle := a|\perp\rangle + \frac{b}{|\mathcal{C}_1|}\sum_{c\in\mathcal{C}_1}|c\rangle$$

for values

$$a := \sqrt{\frac{2\epsilon|\mathcal{C}|}{1+2(|\mathcal{C}|-1)\epsilon}} \quad b := \sqrt{\frac{(1-2\epsilon)|\mathcal{C}|}{1+2(|\mathcal{C}|-1)\epsilon}}$$

It is readily confirmed that $|s\rangle$ is normalized. Moreover,

$$\begin{aligned}
Q|s\rangle &= a/2|\perp\rangle + \frac{a\sqrt{2\epsilon}}{2|\mathcal{C}_1|}\sum_{c\in\mathcal{C}_1}|\phi_c\rangle + \frac{b}{2|\mathcal{C}_1|}\sum_{c\in\mathcal{C}_1}|c\rangle + \frac{b\sqrt{1-2\epsilon}}{2|\mathcal{C}_1|^2}\sum_{c\in\mathcal{C}_1}|\phi_c\rangle \\
&= a/2|\perp\rangle + \frac{b}{2|\mathcal{C}_1|}\sum_{c\in\mathcal{C}_1}|c\rangle + \left(\frac{a\sqrt{2\epsilon}}{2|\mathcal{C}_1|} + \frac{b\sqrt{1-2\epsilon}}{2|\mathcal{C}_1|^2}\right)\sum_{c\in\mathcal{C}_1}(\sqrt{2\epsilon}|\perp\rangle + \sqrt{1-2\epsilon}|c\rangle) \\
&= \left(\frac{a}{2} + a\epsilon + \frac{b\sqrt{1-2\epsilon}\sqrt{2\epsilon}}{2|\mathcal{C}_1|}\right)|\perp\rangle \\
&\quad + \left(\frac{a\sqrt{1-2\epsilon}\sqrt{2\epsilon}}{2|\mathcal{C}_1|} + \frac{b}{2|\mathcal{C}_1|} + \frac{b(1-2\epsilon)}{2|\mathcal{C}_1|^2}\right)\sum_{c\in\mathcal{C}_1}|c\rangle \\
&= \left(\frac{1}{2} + \epsilon + \frac{1-2\epsilon}{2|\mathcal{C}_1|}\right)|s\rangle
\end{aligned}$$

Therefore, $|s\rangle$ is an eigenvector of $\mathcal{P}$ with eigenvalue $\frac{1}{2} + \epsilon + \frac{1-2\epsilon}{2|\mathcal{C}_1|}$. Now imagine running the projective implementation of $\mathcal{P}$. The probability this eigenvalue is found is $|\langle s|\perp\rangle|^2 = a^2$. By our

assumption that the ciphertexts have negligible guessing probability, $|\mathcal{C}_1|$ is super-polynomial. For an inverse polynomial ϵ, this means $a^2 \geq 1-\mathsf{negl}$. The eigenvalue in this case is $\lambda := \frac{1}{2}+\epsilon+\frac{1-2\epsilon}{2|\mathcal{C}_1|} > 1/2+\epsilon$. Thus, if the outcome of the projective implementation is λ, Live happens. Thus $\Pr[\mathsf{Live}] \geq 1-\mathsf{negl}$ as desired. □

We now consider making *classical* Black Box Projection queries to this decoder.

Claim 5.3. *For any j and for any sequence of ciphertexts queries $c_1, \ldots, c_t$ (possibly repeating) to the decoder in the Black Box Projection model, with probability at least $(1-2\epsilon)/(1+2(t-1)\epsilon)$, all outputs of the decoder will be 0.*

Proof. Consider c_i. Suppose all previous queries have resulted in an output of 0. Then at the end of the previous query $\mathcal{H}_2'$ is exactly $|0\rangle$, since it was the result of the previous query's output measurement. If c_i encrypts m_0, then with probability 1 $\mathcal{H}_2'$ will not be affected and the output measurement on query c_i will be 0. Recall that the Black Box Projection query finishes by running $U(c)^\dagger$. In the case c_i encrypts m_0 and all prior queries resulted in 0, the query will thus have no effect on the decoder's state.

Thus, let $d_1, \ldots, d_{t'}$ be the subsequence of $c_1, \ldots, c_t$ consisting only of ciphertexts encrypting m_1. The probability all queries result in 0 for $d_1, \ldots, d_{t'}$ is therefore identical to the probability of all 0's in $c_1, \ldots, c_t$.

Now consider d_i, and suppose all prior queries resulted in 0. We claim this means $\mathcal{H}_2$ is also exactly $|0\rangle$. Indeed, this is true for $i = 1$. Inductively assume this is true for $i-1$. Then when applying the query on d_{i-1}, the contents of $\mathcal{H}_2$ and $\mathcal{H}_2'$ will have the same value in the computational basis, prior to measuring. Thus once we measure $\mathcal{H}_2'$ as part of the Black Box Projection query and get a 0, $\mathcal{H}_2$ will also become 0.

For a ciphertext c, let P_c be the projection $\mathbf{I} - |\phi_c\rangle\langle\phi_c|$ onto the hyperplane orthogonal to $|\phi_c\rangle$. Without re-normalizing, after querying on $d_1, \ldots, d_{t'}$ and conditioning on all the measurement outcomes being 0, the decoder's state is then

$$|\gamma_{t'}\rangle := P_{d_{t'}} \cdot P_{d_{t'-1}} \cdots P_{d_2} \cdot P_{d_1} |\perp\rangle$$

The probability of obtaining all 0's is exactly the norm squared of this un-normalized state.Notice that all the hyperplanes have a common vector:

$$|\eta\rangle := \sqrt{\frac{1-2\epsilon}{1+2(t'-1)\epsilon}}|\perp\rangle - \sqrt{\frac{2\epsilon}{1+2(t'-1)\epsilon}} \sum_{u=1}^{t'} |d_u\rangle$$

Therefore $\langle\eta|\gamma_{t'}\rangle = \langle\eta|\perp\rangle = \sqrt{\frac{1-2\epsilon}{1+2(t'-1)\epsilon}}$. Note that $|\eta\rangle$ is normalized. This means that $|\gamma_{t'}\rangle$ has norm at least $\sqrt{\frac{1-2\epsilon}{1+2(t'-1)\epsilon}}$. Thus, the probability of obtaining all 0's is at least $\frac{1-2\epsilon}{1+2(t'-1)\epsilon} \geq \frac{1-2\epsilon}{1+2(t-1)\epsilon}$, completing the proof of Claim 5.3. □

Now consider any efficient tracing algorithm in the classical Black Box Projection model, and consider its behavior when all of its queries result in 0. If S' is empty in this case, then by Claim 5.3,

$$\Pr[\mathsf{GoodTrace}] \leq 1 - \frac{1-2\epsilon}{1+2(t-1)\epsilon} \ ,$$

which is non-negligibly smaller than $\Pr[\mathsf{Live}] = 1-\mathsf{negl}$, contradicting security.

On the other hand, suppose S' is non-empty in the case of all zeros. Since in this case the view of Trace is independent of j, S' will contain a user other than j with probability at least $1 - 1/N$. Overall, this means Trace accuses an honest user (and hence $\mathsf{BadTrace}$ happens) with probability at least $(1 - 1/N)(1 - 2\epsilon)/(1 + 2(t-1)\epsilon)$, which is non-negligible, again contradicting security. Thus, regardless of how Trace behaves, security is violated. □

6 On Mixtures of Projective Measurements

We now develop some additional tools that will be used in our quantum tracing algorithm in Section 7. We will explore efficient approximations of projective implementations, as well as questions of computational indistinguishability.

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

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

In this section, we will address two questions:

- Since $\mathcal{P}_D$ has a binary outcome, there exists a projective implementation $\mathcal{M} = \mathsf{ProjImp}(\mathcal{P}_D)$. Can we efficiently approximate the measurement?
- If D_0, D_1 are computationally indistinguishable, what does that say about the outcomes of $\mathcal{M}_0 = \mathsf{ProjImp}(\mathcal{P}_{D_0})$ and $\mathcal{M}_1 = \mathsf{ProjImp}(\mathcal{P}_{D_1})$?

6.1 Additional Definitions

Shift distance. 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:

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

Note that small shift distance does *not* imply small statistical difference, as distributions with disjoint supports can have small shift distance. Also note the triangle-like inequality $\Delta^{\epsilon_1+\epsilon_2}_{\mathsf{Shift}}(D_0, D_2) \leq \Delta^{\epsilon_1}_{\mathsf{Shift}}(D_0, D_1) + \Delta^{\epsilon_2}_{\mathsf{Shift}}(D_1, D_2)$.

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

$$\Delta^{\epsilon}_{\mathsf{Shift}}(\mathcal{M},\mathcal{N}) := \sup_{|\psi\rangle} \Delta^{\epsilon}_{\mathsf{Shift}}(\ \mathcal{M}(|\psi\rangle)\ ,\ \mathcal{N}(|\psi\rangle)\ )$$

Almost Projective Measurements. We define "almost" projectivity, based on the fact that repeated consecutive projective measurements yield the same output.

Definition 6.1. *A real-valued quantum measurement $\mathcal{M} = (M_i)_{i\in\mathcal{I}}$ is (ϵ,δ)-almost projective if the following is true: for any quantum state $|\psi\rangle$, apply $\mathcal{M}$ twice in a row to $|\psi\rangle$, obtaining measurement outcomes x, y. Then $\Pr[|x-y|\leq\epsilon] \geq 1-\delta$.*

6.2 Approximating Projective Implementations

We now address the question of efficiently approximating the projective implementation $\mathcal{M} = \mathsf{ProjImp}(\mathcal{P}_D)$ of a mixture of projective measurements $\mathcal{P}_D$. We note that exact measurement is computationally infeasible, as it captures computing acceptance probabilities of circuits. Instead, we employ techniques from [MW04] to develop an algorithm API which efficiently *approximates* the projective implementation of $\mathcal{P}_D$. We first define two subroutines.

Controlled Projection. Let $\mathcal{P} = \{\mathcal{P}_i = (P_i, Q_i)\}_{i\in\mathcal{I}}$ be a collection of projective measurements over $\mathcal{H}$. Let D a distribution with random coin set $\mathcal{R}$. We will abuse notation and let $\mathcal{R}$ also denote the $|\mathcal{R}|$-dimensional Hilbert space. The *controlled projection* is the measurement $\mathsf{CProj}_{\mathcal{P},D} = (\mathsf{CProj}^0_{\mathcal{P},D}, \mathsf{CProj}^1_{\mathcal{P},D})$ where

$$\mathsf{CProj}^0_{\mathcal{P},D} = \sum_{r\in\mathcal{R}} |r\rangle\langle r| \otimes P_{D(r)} \quad , \quad \mathsf{CProj}^1_{\mathcal{P},D} = \sum_{r\in\mathcal{R}} |r\rangle\langle r| \otimes Q_{D(r)} \ .$$

$\mathsf{CProj}_{\mathcal{P},D}$ is readily implemented using the measurement in Equation 1. First, initialize control registers $\mathcal{I}$ to 0. Then perform the map $|r\rangle|i\rangle \mapsto |r\rangle|i \oplus D(r)\rangle$ to the $\mathcal{R}\times\mathcal{I}$ registers. Next, apply the mixture of projective measurements assumed in Equation 1. Finally, perform the map $|r\rangle|i\rangle \mapsto |r\rangle|i\oplus D(r)\rangle$ again to un-compute the control registers, and discard the control registers.

Uniform Test. Define $\mathsf{IsUniform}_{\mathcal{R}} = (|\mathbb{1}_{\mathcal{R}}\rangle\langle\mathbb{1}_{\mathcal{R}}|, \mathbf{I} - |\mathbb{1}_{\mathcal{R}}\rangle\langle\mathbb{1}_{\mathcal{R}}|)$ where

$$|\mathbb{1}_{\mathcal{R}}\rangle = \frac{1}{\sqrt{|\mathcal{R}|}} \sum_{r\in\mathcal{R}} |r\rangle \ .$$

The Algorithm API. Our algorithm is parameterized by a distribution D, collection of projective measurements $\mathcal{P}$, and real values $0 < \epsilon, \delta \leq 1$, and is denoted as $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$. On input a quantum state $|\psi\rangle$ over Hilbert space $\mathcal{H}$, it works as follows:

1. Initialize a new register $\mathcal{R}$ to the state $|\mathbb{1}_{\mathcal{R}}\rangle$.
2. Initialize a classical list $L = (0)$.
3. Repeat the following "main loop" a total of $T = \lceil \ln(4/\delta)/\epsilon^2 \rceil$ times:

(a) Apply the controlled projection $\mathsf{CProj}_{\mathcal{P},D}$ over the joint system $\mathcal{R} \otimes \mathcal{H}$, resulting in measurement outcome b_{2i-1}. Append b_{2i-1} to the end of L.

(b) Apply the Uniform Test $\mathsf{IsUniform}_{\mathcal{R}}$ to the system $\mathcal{R}$, resulting in measurement outcome b_{2i}. Append b_{2i} to the end of L.

4. Let t be the number of bit flips in the sequence $L = (0, b_1, b_2, \ldots, b_{2T})$, and let $\tilde{p} = t/2T$ be the fraction of bit flips

5. If in the last iteration of the "main loop" $b_{2T} = 1$, repeat the "main loop" until the first time $b_{2i} = 0$.

6. Discard the $\mathcal{R}$ registers, and output $\tilde{p}$.

Theorem 6.2. *For any $\epsilon, \delta, \mathcal{P}, D$, we have that:*

- $\Delta^{\epsilon}_{\mathsf{Shift}}(\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}, \mathsf{ProjImp}(\mathcal{P}_D)) \leq \delta$. *That is,* API *approximates the projective implementation* $\mathsf{ProjImp}(\mathcal{P}_D)$.

- $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ *is (ϵ, δ)-almost projective.*

- *The expected run time of* $\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D}$ *is $X \times \mathsf{poly}(1/\epsilon, \log(1/\delta))$, where X is the combined run time of D, the procedure mapping i to the measurement (P_i, Q_i), and the run-time of the measurement (P_i, Q_i).*

Proof. Let $|\psi\rangle$ be an arbitrary state. Write $|\psi\rangle = \sum_p \alpha_p |\psi_p\rangle$ where $|\psi_p\rangle$ are eigenvectors of $\mathcal{P}_D$ with eigenvalue p [7]. In other words, $Q_D|\psi_p\rangle = p|\psi_p\rangle$. Define the following states:

- $|u^0_p\rangle = \frac{1}{\sqrt{(1-p)|\mathcal{R}|}} \sum_r |r\rangle P_{D(r)}|\psi_p\rangle$. Notice that

$$\begin{aligned}\langle u^0_p|u^0_p\rangle &= \frac{1}{(1-p)|\mathcal{R}|}\left(\sum_r \langle r|\langle\psi_p|P_{D(r)}\right)\left(\sum_s |s\rangle P_{D(s)}|\psi_p\rangle\right) \\ &= \frac{1}{(1-p)}\langle\psi_p|\left(\frac{1}{|\mathcal{R}|}\sum_r P_{D(r)}\right)|\psi_p\rangle = \frac{1}{1-p}\langle\psi_p|P_D|\psi_p\rangle = 1 \ .\end{aligned}$$

  Also, notice that $\mathsf{CProj}^0_{\mathcal{P},D}|u^0_p\rangle = |u^0_p\rangle$ whereas $\mathsf{CProj}^1_{\mathcal{P},D}|u^0_p\rangle = 0$.

- $|u^1_p\rangle = \frac{1}{\sqrt{p|\mathcal{R}|}} \sum_r |r\rangle Q_{D(r)}|\psi_p\rangle$. By an analogous calculation for $|u^0_p\rangle$, we have that $\langle u^1_p|u^1_p\rangle = 1$. Since different eigenvectors of $\mathcal{P}_D$ are orthogonal, we also have that $\langle u^1_p|u^1_{p'}\rangle = 0$ for $p \neq p'$. Since $P_i Q_i = 0$, we have $\langle u^1_p|u^0_{p'}\rangle = 0$ for any p, p' (not necessarily distinct). This means $B = \{|u^b_p\rangle\}_{b,p}$ is orthonormal. Also, notice that $\mathsf{CProj}^0_{\mathcal{P},D}|u^1_p\rangle = 0$ whereas $\mathsf{CProj}^1_{\mathcal{P},D}|u^1_p\rangle = |u^1_p\rangle$.

- $|v^0_p\rangle = |\mathbb{1}_{\mathcal{R}}\rangle \otimes |\psi_p\rangle$. Notice that $|v^0_p\rangle = \sqrt{1-p}|u^0_p\rangle + \sqrt{p}|u^1_p\rangle$. Also notice that $\mathsf{IsUniform}^0_{\mathcal{R}} \otimes \mathbf{I}|v^0_p\rangle = |v^0_p\rangle$ and $\mathsf{IsUniform}^1_{\mathcal{R}} \otimes \mathbf{I}|v^0_p\rangle = 0$

- $|v^1_p\rangle = -\sqrt{p}|u^0_p\rangle + \sqrt{1-p}|u^1_p\rangle$. Notice that $\langle v^b_p|v^{b'}_{p'}\rangle$ is 1 if $b = b' \wedge p = p'$ and 0 otherwise. This means $B' = \{|v^b_p\rangle\}$ is orthonormal, spanning the same space as B. Finally, notice that $\mathsf{IsUniform}^0_{\mathcal{R}} \otimes \mathbf{I}|v^1_p\rangle = 0$ and $\mathsf{IsUniform}^1_{\mathcal{R}} \otimes \mathbf{I}|v^1_p\rangle = |v^1_p\rangle$.

[7] Note that there may be repeated eigenvalues. The $|\psi_p\rangle$ are therefore the projections of $|\psi\rangle$ onto the eigenspaces.

At the beginning of the first run of the "main loop" (Step 3), the state of the system is $|\psi_\emptyset\rangle := |\mathbb{1}_{\mathcal{R}}\rangle \otimes |\psi\rangle$. Writing this state in the basis B', we have

$$|\mathbb{1}_{\mathcal{R}}\rangle \otimes |\psi\rangle = \sum_p \alpha_p |v_p^0\rangle \ .$$

Let $|\psi_L\rangle$ for $L \in \{0,1\}^z$ denote the *unnormalized* state of the system after the first z measurements, if the sequence of measurement outcomes is L. Let $t(L)$ denote the number of bit flips in the sequence $0, L_1, L_2, \ldots, L_z$.

Claim 6.3. $|\psi_L\rangle = \theta_L \sum_p \alpha_p (\sqrt{p})^{t(L)} (\sqrt{1-p})^{z-t(L)} \begin{cases} |v_p^{L_z}\rangle & \text{if } z \bmod 2 = 0 \\ |u_p^{L_z}\rangle & \text{if } z \bmod 2 = 1 \end{cases}$ *where* θ_L *is a global phase factor,* $|\theta_L| = 1$.

Proof. We prove by induction. The base case $z = 0$ is true. Now assume that the claim is true for $z-1$. We prove the odd z case, the even case being essentially identical. Let L' be L but with the last entry removed. By induction we have

$$|\psi_{L'}\rangle = \theta_{L'} \sum_p \alpha_p (\sqrt{p})^{t(L')} (\sqrt{1-p})^{z-1-t(L')} |v_p^{L'_z}\rangle \ .$$

Observe that $|v_p^b\rangle = \sqrt{1-p}|u_p^b\rangle - (-1)^b |u_b^{1-b}\rangle$. We apply $\mathsf{CProj}_{\mathcal{P},D}$; if the outcome is b, this projects onto $\{|u_p^b\rangle\}_p$. If $L_z = L'_{z-1} \oplus c$, then $t(L) = t(L') + c$, and

$$|\psi_L\rangle = \theta_{L'} (-1)^{cL_z} \sum_p \alpha_p (\sqrt{p})^{t(L')+c} (\sqrt{1-p})^{z-t(L')-c} |u_p^{L_z}\rangle$$

Setting θ_L appropriately gives the desired outcome. □ □

At Step 4, the unnormalized state is $|\psi_L\rangle$ as defined above, where L contains the results of measurements. The probability of obtaining a particular L is

$$\langle \psi_L | \psi_L \rangle = \sum_p |\alpha_p|^2 (p)^{t(L)} (1-p)^{2T-t(L)} \ .$$

L is therefore distributed according to the following distribution:

- First apply $\mathsf{ProjImp}(\mathcal{P}_D)$ to $|\psi\rangle$ to obtain a value p
- Let K be a list of $2T$ independent coin flips with expected value p.
- Set L_i to be the parity of the first i bits of K.

Then $2T\tilde{p} = t(L)$ is just the number 1s in K. Hoeffding's inequality then gives

$$\Pr[|\tilde{p} - p| \geq \epsilon/2] \leq 2e^{-2(2T)(\epsilon/2)^2} \leq \delta/2 < \delta \ ,$$

for $T \geq \ln(4/\delta)/\epsilon^2$. This implies that $\Delta^{\epsilon}_{\mathsf{Shift}}(p, \tilde{p}) \leq \Delta^{\epsilon/2}_{\mathsf{Shift}}(p, \tilde{p}) \leq \delta/2 \leq \delta$.

We now analyze the run-time, which is dominated by the number of iterations of the main loop, including Step 5. Note that Step 5 terminates once the number of bit flips in L is even. The number of iterations is identically distributed to:

- Sample p by running $\mathsf{ProjImp}(\mathcal{P}_D)$.
- Flip $2T$ biased random coins whose probability of outputting 1 is p.
- Flip an even number of additional coins until the overall parity is 0.
- Output the total number of coin tosses, divided by 2.

We can simplify this experiment by pairing off the coin tosses, and only looking at the parity of each pair, which itself is a biased coin with expectation $q = 2p(1-p)$:

- Sample p by running $\mathsf{ProjImp}(\mathcal{P}_D)$.
- Flip T biased random coins whose probability of outputting 1 is $q = 2p(1-p)$.
- Flip additional coins until the overall parity is 0.
- Output the total number of coin tosses.

Let $T'(q)$ be the expected number of additional coins for a given q; note that $q \in [0, 1/2]$. Note that $T'(0) = 0$, since the parity is always even. For $q > 0$, if the parity is even after T steps, no additional flips are needed. Assuming T is even, a routine calculation shows that the probability the parity is odd after the first T steps is $(1-(1-2q)^{2T})/2$, in which case an expected $1/q$ additional flips are needed. Thus $T'(q) := (1-(1-2q)^{2T})/2q$ for $q > 0$. For $q \in (0, 1/2]$, T' is monotonically decreasing, and $\lim_{q\to 0} T'(q) = 2T$. Therefore, for any fixed q, we can upper bound the total expected number of coin tosses to $T + 2T = 3T$. By linearity of expectation, this also holds over any distribution over q. Thus, the expected number of runs of the main loop is at most $3T$.

Finally, we consider applying API twice to the same state. Notice that, since the first run of API is guaranteed to stop when the last bit of L is 0, this corresponds to $\mathcal{R}$ containing a uniform superposition. But this means that when we start the second run of API, the state going into the main loop will actually be identical to the state at the end of the first run. We can therefore view the two runs of API as a single run, but with a larger value of T. The overall list K produced by both runs, but stopping at Step 4 in the second run, is then distributed according to:

- Sample p by running $\mathsf{ProjImp}(\mathcal{P}_D)$.
- Flip $2T$ biased random coins whose probability of outputting 1 is p.
- Flip an even number of additional random coins, until a 0 is found.
- Then flip $2T$ more biased random coins.
- Let K be the overall list of coin flips.

The first output, $\tilde{p}_1$, is then just the fraction of 1's in the first $2T$ bits of K, whereas the second output, $\tilde{p}_2$, is the fraction of 1's in the last $2T$ bits of K. These fractions are independent. Recalling that

$$\Pr[|\tilde{p} - p| \geq \epsilon/2] \leq \delta/2,$$

we have that $\Pr[|\tilde{p}_1 - \tilde{p}_2| \geq \epsilon] \leq \delta$. Thus API is (ϵ, δ)-almost projective. □ □

Remark 6.4. *Note that we only bound the* expected *runtime of* API *to be a polynomial. One may hope to give a worst-case polynomial bound, by cutting off Step 5 after a given number of iterations. Unfortunately, the number of iterations will have to depend on p. Indeed, for any polynomial upper bound on the number of iterations in Step 5, for p very close to 0 or 1 (and hence q very close to 0), there will be some small-but-non-negligible probability of having a single bit flip in the first $2T$ steps, but then having no further bit flips in the truncated Step 5.*

We note that if p is guaranteed to be far from 0 or 1 (say, $p \in [1/4, 3/4]$), then we can *truncate Step 5 and incur only a negligible failure probability. One can in turn guarantee that p is far from 0 or 1 by adding dummy projections to $\mathcal{P}$ that always output 0 and also projections that always output 1. Once $\tilde{p}$ is computed for this modified $\mathcal{P}$, an estimate for the original $\mathcal{P}$ can be obtained by scaling. Thus, it is also possible to design an algorithm with strict polynomial run-time, though it will be somewhat more complicated. For ease of exposition, we opted for the simpler algorithm that only achieves expected polynomial time.*

6.3 On Computational Indistinguishability

Here, we show that if the underlying distributions D_0, D_1 are computationally indistinguishable, then the resulting projective implementations $\mathcal{M}_0 = \mathsf{ProjImp}(\mathcal{P}_{D_0})$ and $\mathcal{M}_1 = \mathsf{ProjImp}(\mathcal{P}_{D_1})$ are close.

Theorem 6.5. *Let ρ be an efficiently constructible mixed state, and D_0, D_1 efficiently sampleable, computationally indistinguishable distributions. For any inverse polynomial ϵ, there exists a negligible δ such that $\Delta^{\epsilon}_{\mathsf{Shift}}(\mathcal{M}_0(\rho), \mathcal{M}_1(\rho)) \leq \delta$.*

Proof. The rough idea is that we will switch from the projective implementation $\mathcal{M}_b$ to our approximation API. Since API is efficient, we argue that the results of API must be close. The difficulty is that API makes queries on a *superposition* of exponentially-many samples from the respective D_b distribution, whose indistinguishability does not follow from the indistinguishability of single samples. We nevertheless show that the outputs of API under the two distributions must be close by using the small-range distributions of Zhandry [Zha12a].

Consider an adversary A producing a mixture ρ. Let $\mathcal{R}$ be the space of random coins for D_0, D_1; we can assume wlog that they share the same random coin space. We now define the following sequence of hybrid distributions:

Hybrid 0. The distribution is $p_0 \leftarrow \mathcal{M}_0(\rho)$ where ρ is generated by A.

Hybrid 1. Here, we choose a random permutation Π on $\mathcal{R}$. Let $D_0^{\Pi}(r) = D_0(\Pi(r))$. Run $p_1 \leftarrow \mathsf{ProjImp}(\mathcal{P}_{D_0^{\Pi}})$. Since D_0 and D_0^{Π} are identical distributions, the measurements $\mathcal{P}_{D_0}$ and $\mathcal{P}_{D_0^{\Pi}}$ are identical, and therefore so are their projective implementations. Thus, p_0 and p_1 are identically distributed.

Hybrid 2. Here, we will generate $p_2 \leftarrow \mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_0^{\Pi}}(\rho)$, for a function δ' and an inverse polynomial ϵ' to be chosen later. By Theorem 6.2, we have that $\Delta^{\epsilon'}_{\mathsf{Shift}}(p_1, p_2) \leq \delta'$.

Hybrid 3. Now we change Π to be the small-range functions $\Sigma = G \circ F$ of Zhandry [Zha12a], where $F : \mathcal{R} \to [s]$ and $G : [s] \to \mathcal{R}$ are random functions, and s is a parameter. Let $p_3 \leftarrow \mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_0^{\Sigma}}(\rho)$. Let Φ be the distribution of random *functions* on $\mathcal{R}$. Yuen and Zhandry show the following:

Theorem 6.6 ([Yue14, Zha15])**.** *For any quantum algorithm B making Q quantum queries to Π or Φ, $|\Pr[B^{\Pi}() = 1] - \Pr[B^{\Phi}() = 1]| \leq O(Q^3/|\mathcal{R}|)$.*

Theorem 6.7 ([Zha12a])**.** *For any quantum algorithm B making Q quantum queries to Φ or Σ, $|\Pr[B^{\Phi}() = 1] - \Pr[B^{\Sigma}() = 1]| \leq O(Q^3/|\mathcal{R}|)$.*

Theorems 6.6 and 6.7 in particular means that $\Delta^0_{\mathsf{Shift}}(p_2, p_3) \leq O(Q^3/s + Q^3/|\mathcal{R}|)$.

Hybrid 4. This is the same as Hybrid 3, except that we change F to be a $2Q$-wise independent function E. Let $p_4 \leftarrow \mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_0^{G\circ E}}(\rho)$. Since API only makes Q queries to F or E, the following theorem implies that p_3 and p_4 are identically distributed:

Theorem 6.8 ([Zha12b])**.** *For any quantum algorithm B making Q quantum queries to F or E, $\Pr[B^F() = 1] = \Pr[B^E() = 1]$.*

Assume $|\mathcal{R}| > s$, adding random coins to $\mathcal{R}$ that are ignored by D_0, D_1 if necessary. Then $\Delta^{\epsilon'}_{\mathsf{Shift}}(p_0, p_4) \leq O(Q^3/s) + \delta'$.

Hybrid 5. Next, we switch to using the distribution $D_1^{G\circ E}(r) = D_1(G(E(r)))$. Let $p_5 \leftarrow \mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_1^{G\circ E}}(\rho)$. Note that $D_b(G(\cdot))$ can be interpreted as a list of s samples from D_b, which the input selecting which sample to use. Since D_0 and D_1 are computationally indistinguishable, so are s samples. Notice that the entire experiment in Hybrids 4/5 are efficient. Therefore, by a straightforward argument, we have that $\Delta^0_{\mathsf{Shift}}(p_5, p_6) \leq \gamma$ where γ is negligible.

Hybrids 6-9. **Hybrid** $6+g$ is identical to **Hybrid** $5-g$ except for replacing D_0 with D_1. In **Hybrid 9**, the output is exactly $\mathcal{M}_1(\rho)$. Putting everything together, we have that $\Delta^{2\epsilon'}_{\mathsf{Shift}}(\mathcal{M}_0(\rho), \mathcal{M}_1(\rho)) \leq O(Q^3/s) + 2\delta' + \gamma$.

Let ϵ be an inverse polynomial, and suppose $\delta := \Delta^{\epsilon}_{\mathsf{Shift}}(\mathcal{M}_0(\rho), \mathcal{M}_1(\rho))$ is non-negligible, lower bounded by an inverse-polynomial w infinitely often. Set $\epsilon' = \epsilon/2$ and $\delta' = w/4$. Then $\log(1/\delta')$ is logarithmic. Recall $Q = O(\log(1/\delta')^3/(\epsilon')^2)$. For the infinitely-many values of the security parameter where $\delta \geq w$, we have that $w \leq \delta \leq O(Q^3/s) + w/2 + \gamma$, which re-arranges to $w \leq O(\log(1/w)^3/\epsilon^6 s) + 2\gamma$. But now choose $s = 2 \times O(\ (1/\epsilon)^6 (1/w) \log(1/w)^3\ )$, a polynomial. This gives $w \leq w/2 + 2\gamma$, or $w \leq 4\gamma$, which can only happen for finitely many security parameters since γ is negligible, a contradiction. Thus δ must be negligible. □

We conclude this section with an immediate corollary of Theorem 6.5:

Corollary 6.9. *Let ρ be an efficiently constructible, potentially mixed state, and let D_0, D_1 be two computationally indistinguishable distributions. Then for any inverse polynomial ϵ and any function δ, there exists a negligible* negl *such that $\Delta^{3\epsilon}_{\mathsf{Shift}}(\mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D_0}, \mathsf{API}^{\epsilon,\delta}_{\mathcal{P},D_1}) \leq 2\delta + \mathsf{negl}$.*

7 Tracing PLBE

7.1 Private Linear Broadcast Encryption

Our construction will use the Private Linear Broadcast Encryption (PLBE) framework of Boneh, Sahai, and Waters [BSW06]. A PLBE scheme is a triple of probabilistic *classical* polynomial time algorithms $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ where:

- $\mathsf{Gen}'(1^N, 1^\lambda)$ takes as input a number of users N and a security parameter λ. It outputs a public key pk, plus N user secret keys sk_i for $i \in [N]$.
- $\mathsf{Enc}'(\mathsf{pk}, j, m)$ takes as input the public key, an index $j \in [0, N]$, and a message m. It outputs a ciphertext c.
- $\mathsf{Dec}'(\mathsf{sk}_i, c)$ takes as input a secret key sk_i for user i and a ciphertext, and outputs a message m' or a special abort symbol $\perp$.

Correctness. For correctness, we require that user i can decrypt ciphertexts with index j, so long as $i \leq j$. That is there exists a negligible function $\mathsf{negl}(\lambda)$ such that for every λ and $N \leq 2^\lambda$, for every $i \in [N]$ and $j \geq i$, we have that

$$\Pr[\mathsf{Dec}'(\mathsf{sk}_i, \mathsf{Enc}'(\mathsf{pk}, j, m)) = m : (\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in[N]}) \leftarrow \mathsf{Gen}'(N, \lambda)] > 1 - \mathsf{negl}(\lambda) \ .$$

Security. We need two security requirements. The first is *indistinguishability security*, which requires semantic security for encryptions to $j = 0$:

Definition 7.1. *A PLBE scheme* $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *is* indistinguishable secure *if for all quantum polynomial time adversaries A, there exists a negligible* negl *such that the probability A wins in the following game is at most* $1/2 + \mathsf{negl}(\lambda)$*:*

- *A gets λ as input, and sends a number N represented in unary.*
- *Run* $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}'(\lambda, N)$*, and send* pk *to A.*
- *A then makes an arbitrary number of classical queries on identities $i \in [N]$; in response it receives* sk_i*.*
- *Next, A outputs a pair of messages (m_0, m_1). In response, choose a random bit b and send A the ciphertext* $c \leftarrow \mathsf{Enc}'(\mathsf{pk}, j = 0, m_b)$*.*
- *A makes more queries for* sk_i*.*
- *Finally, A outputs a guess b' for b. Output "win" if and only if $b' = b$.*

Second, we need *index hiding security* which says that encrypts to $j - 1$ and j are only distinguishable to an adversary that has the secret key for user j.

Definition 7.2. *A PLBE scheme* $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *is* index hiding secure *if for all quantum polynomial time adversaries A, there exists a negligible function* negl *such that the probabilities A wins in the following game is at most* $1/2 + \mathsf{negl}(\lambda)$*:*

- *A gets λ as input, and sends a number N represented in unary.*
- *Run $(\mathsf{pk}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Gen}'(\lambda, N)$, and send pk to A.*
- *A then makes an arbitrary number of classical queries on identities $i \in [N]$; in response it receives sk_i. Let S be the set of i queried by A.*
- *Next, A outputs a pair of (j, m) for $j \in [N]$ such that $j \notin S$. Choose a random bit b and send A the ciphertext $c \leftarrow \mathsf{Enc}'(\mathsf{pk}, j - b, m)$ to index $j - b$*
- *A is allowed to make more queries on identities $i \in [N] \setminus j$, to which it receives sk_i in response.*
- *Finally, A outputs a guess b' for b. Output "win" if and only if $b' = b$.*

From PLBE to Traitor Tracing. Following [BSW06], the first three algorithms of our traitor tracing construction $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ we be immediately derived from the PLBE scheme: $\mathsf{Gen} = \mathsf{Gen}'$, $\mathsf{Enc}(\mathsf{pk}, m) = \mathsf{Enc}'(\mathsf{pk}, j = N, m)$, and $\mathsf{Dec} = \mathsf{Dec}'$. Correctness is immediate. In the following, we describe Trace.

7.2 The quantum algorithm Trace.

Where we depart from [BSW06] is in our tracing algorithm, which we now need to trace quantum pirates. First, we briefly explain how to implement API using Black Box Projection queries.

Concretely, let $|☠'\rangle$ be $|☠\rangle$, except that we augment the decoder with a qubit $\mathcal{H}_2$ originally set to $|0\rangle$. Let $\mathcal{H}_2' \times \mathcal{C}$ be control registers, where $\mathcal{H}_2'$ is another qubit and $\mathcal{C}$ is a ciphertext register. Consider the following measurement process on registers $\mathcal{H}_2' \otimes \mathcal{C} \otimes \mathcal{H}_2 \otimes \mathcal{H}$:

- Perform the map $|b'\rangle|b\rangle \to |b'\rangle|b \oplus b'\rangle$ on the $\mathcal{H}_2' \otimes \mathcal{H}_2$ registers
- Make a Black Box Projection query using the registers $\mathcal{C} \otimes \mathcal{H}_2$ as the query registers. Let o be the result.
- Perform the map $|b'\rangle|b\rangle \to |b'\rangle|b \oplus b'\rangle$ on the $\mathcal{H}_2' \otimes \mathcal{H}_2$ registers
- Output $1 - o$.

This measurement process has exactly the form of a collection of projective measurements $\mathcal{P}$ in Equation 1. For a decoder in its initial state (meaning $\mathcal{H}_2$ is initialized to $|0\rangle$) and for a given bit/ciphertext pair (b, c), the corresponding measurement $\mathcal{P}_{(b,c)}$ outputs 1 exactly when the decoder would output b. Thus, we can run the algorithm API on $|☠'\rangle$.

We now give our algorithm $\mathsf{Trace}^{|☠\rangle}(\mathsf{pk}, m_0, m_1, \epsilon)$:

1. Let $\epsilon' = \epsilon/4(N+1)$ and $\delta' = 2^{-\lambda}$.
2. Run $\tilde{p}_N \leftarrow \mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_N}(|☠'\rangle)$, where D_j is the following distribution:
   - Run $b \leftarrow \{0, 1\}$
   - Compute $c \leftarrow \mathsf{Enc}'(\mathsf{pk}, j, m_b)$
   - Output (b, c).

3. If $\tilde{p}_N < 1/2 + \epsilon - \epsilon'$, abort and output the empty set $\{\}$.

4. Otherwise, initialize $S' = \{\}$. Then for $j = N$ to $j = 1$,

   - Compute $\tilde{p}_{j-1} \leftarrow \mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_{j-1}}(|\text{☠}'\rangle)$
   - If $\tilde{p}_{j-1} < \tilde{p}_j - 4\epsilon'$, add j to S'.

   Finally, output S'.

Theorem 7.3. *If* $(\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ *is indistinguishable secure and index hiding secure for quantum adversaries, then* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *is quantum traceable.*

Proof. Consider an adversary A which has secret keys for identities in S, and produces a pirate decoder $|\text{☠}\rangle$. Let ϵ be an inverse polynomial. Define the events $\mathsf{GoodTrace}, \mathsf{BadTrace}, \mathsf{Live}$ as in Definition 4.4.

We first argue that $\Pr[\mathsf{BadTrace}]$ is negligible. Suppose that there is a non-negligible probability s that $\mathsf{BadTrace}$ happens. Then for a random choice of j, it is the case that with (non-negligible) probability at least s/N, both (1) A never queries j, and (2) $\tilde{p}_{j-1} < \tilde{p}_j - 4\epsilon'$.

Let ρ be the state produced by the following process:

- Choose a random j, and run the tracing experiment
- If A ever makes a query on j, abort and output an arbitrary quantum state.
- Next run Trace, stopping immediately after $\tilde{p}_j$ is computed.
- Output the state $|\text{☠}'\rangle$.

Consider running Trace for one more iteration, applying $\mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_{j-1}}$ to ρ to obtain a measurement $\tilde{p}_{j-1}$. By assumption, we have that $\tilde{p}_{j-1} \geq \tilde{p}_j - 4\epsilon'$ with non-negligible probability s/N.

Now consider instead stopping Trace at iteration j to obtain ρ, but then applying $\mathsf{API}^{\epsilon',\delta'}_{\mathcal{P},D_j}$ to ρ a second time, obtaining a second measurement $\tilde{p}'_j$ of p_j. We stress that in this case, we do *not* compute $\tilde{p}_{j-1}$. Since API is (ϵ', δ') projective, we know that $|\tilde{p}_j - \tilde{p}'_j| \leq \epsilon'$ except with probability at most δ'.

Since j was never queried, encryptions to index j and $j-1$ are indistinguishable. By Corollary 6.9, this means the distributions on $\tilde{p}'_j$ and $\tilde{p}_{j-1}$ satisfy $\Delta^{3\epsilon'}_{\mathsf{Shift}}(\tilde{p}'_j, \tilde{p}_{j-1}) \leq \mathsf{negl}$. But by our triangle-like inequality, this means that $\tilde{p}_{j-1} \geq \tilde{p}_j - 4\epsilon'$ except with negligible probability, a contradiction.

We now argue that $\Pr[\mathsf{GoodTrace}] \geq \Pr[\mathsf{Live}] - \mathsf{negl}(\lambda)$. First, let Abort be the event that tracing aborts in Step 3. Let p_N be the probability obtained from applying $\mathcal{M}'$ to the decoder outputted by A. Note that Live is the event that $p_N > 1/2 + \epsilon$. We then have that $\Delta^{\epsilon'}_{\mathsf{Shift}}(p_N, \tilde{p}_N) \leq \delta'$. Therefore, $\Pr[\neg\mathsf{Abort}] \geq \Pr[\mathsf{Live}] - \delta'$

Next, let Fail be the event that $\tilde{p}_0 \geq 1/2 + 4\epsilon'$. Let ρ be the state right before measuring $\tilde{p}_0$. Let p_0 be the random variable corresponding to applying $\mathcal{P}_{D_0}$ to ρ. Recall that for $j = 0$, encryptions of m_0 and m_1 are computationally indistinguishable. This means that $p_0 \leq 1/2 + \mathsf{negl}$. By Corollary 6.9, this means $\Pr[\mathsf{Fail}] < \mathsf{negl}$. Thus, $\Pr[\neg\mathsf{Abort} \wedge \neg\mathsf{Fail}] \geq \Pr[\mathsf{Live}] - \mathsf{negl}$.

Finally, we note that if neither of Fail or Abort happen, then $\tilde{p}_N - \tilde{p}_0 > \epsilon - 4\epsilon' = 4N\epsilon'$. But then it must have been some j such that $\tilde{p}_j - \tilde{p}_{j-1} > 4\epsilon'$, meaning S' is non-empty and therefore $\mathsf{GoodTrace}$ happens. Thus $\Pr[\mathsf{GoodTrace}] \geq \Pr[\neg\mathsf{Abort} \wedge \neg\mathsf{Fail}] \geq \Pr[\mathsf{Live}] - \mathsf{negl}$, as desired. □

References

[Aar04] Scott Aaronson. Limitations of quantum advice and one-way communication. In *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004.*, pages 320–332. IEEE, 2004.

[Aar09] S. Aaronson. Quantum copy-protection and quantum money. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 229–242, 2009.

[AC12] Scott Aaronson and Paul Christiano. Quantum money from hidden subspaces. In Howard J. Karloff and Toniann Pitassi, editors, *44th ACM STOC*, pages 41–60. ACM Press, May 2012.

[AGKZ20] Ryan Amos, Marios Georgiou, Aggelos Kiayias, and Mark Zhandry. One-shot signatures and applications to hybrid quantum/classical authentication. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 255–268. ACM Press, June 2020.

[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.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part I*, volume 11891 of *LNCS*, pages 174–198. Springer, Heidelberg, December 2019.

[BB87] Charles H. Bennett and Gilles Brassard. Quantum public key distribution reinvented. *SIGACT News*, 18(4):51–53, July 1987.

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In Mikkel Thorup, editor, *59th FOCS*, pages 320–331. IEEE Computer Society Press, October 2018.

[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.

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 544–574. Springer, Heidelberg, November 2018.

[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.

[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.

[BZ14] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014.

[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.

[CHN+16] Aloni Cohen, Justin Holmgren, Ryo Nishimaki, Vinod Vaikuntanathan, and Daniel Wichs. Watermarking cryptographic capabilities. In Daniel Wichs and Yishay Mansour, editors, *48th ACM STOC*, pages 1115–1127. ACM Press, June 2016.

[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.

[GKRW18] Rishab Goyal, Venkata Koppula, Andrew Russell, and Brent Waters. Risky traitor tracing and new differential privacy negative results. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 467–497. Springer, Heidelberg, August 2018.

[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.

[GKZ19] Alex B. Grilo, Iordanis Kerenidis, and Timo Zijlstra. Learning-with-errors problem is easy with quantum samples. *Phys. Rev. A*, 99:032314, Mar 2019.

[KLLN16] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and María Naya-Plasencia. Breaking symmetric cryptosystems using quantum period finding. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part II*, volume 9815 of *LNCS*, pages 207–237. Springer, Heidelberg, August 2016.

[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. IEEE, 2010.

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

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

[VDG98] Jeroen Van De Graaf. Towards a formal definition of security for quantum protocols, 1998.

[Wie83] Stephen Wiesner. Conjugate coding. *SIGACT News*, 15(1):78–88, January 1983.

[Yue14] Henry Yuen. A quantum lower bound for distinguishing random functions from random permutations. *Quantum Information & Computation*, 14(13-14):1089–1097, 2014.

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

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

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