Succinct Classical Verification of Quantum Computation

James Bartusek* Yael Tauman Kalai† Alex Lombardi‡ Fermi Ma§
Giulio Malavolta¶ Vinod Vaikuntanathan‖ Thomas Vidick** Lisa Yang††

June 28, 2022

Abstract

We construct a classically verifiable *succinct* interactive argument for quantum computation (BQP) with communication complexity and verifier runtime that are poly-logarithmic in the runtime of the BQP computation (and polynomial in the security parameter). Our protocol is secure assuming the post-quantum security of indistinguishability obfuscation (iO) and Learning with Errors (LWE). This is the first succinct argument for quantum computation *in the plain model*; prior work (Chia-Chung-Yamakawa, TCC '20) requires both a long common reference string and non-black-box use of a hash function modeled as a random oracle.

At a technical level, we revisit the framework for constructing classically verifiable quantum computation (Mahadev, FOCS '18). We give a self-contained, modular proof of security for Mahadev's protocol, which we believe is of independent interest. Our proof readily generalizes to a setting in which the verifier's first message (which consists of many public keys) is *compressed*. Next, we formalize this notion of compressed public keys; we view the object as a generalization of constrained/programmable PRFs and instantiate it based on indistinguishability obfuscation.

Finally, we compile the above protocol into a fully succinct argument using a (sufficiently composable) succinct argument of knowledge for NP. Using our framework, we achieve several additional results, including

- Succinct arguments for QMA (given multiple copies of the witness),
- Succinct *non-interactive* arguments for BQP (or QMA) in the quantum random oracle model, and
- Succinct batch arguments for BQP (or QMA) assuming post-quantum LWE (without iO).

*UC Berkeley. Email: `bartusek.james@gmail.com`.

†Microsoft Research and MIT. Email: `yael@microsoft.com`.

‡MIT. Email: `alexjl@mit.edu`.

§Simons Institute and UC Berkeley. Email: `fermima@alum.mit.edu`.

¶Max Planck Institute for Security and Privacy. Email: `giulio.malavolta@hotmail.it`.

‖MIT. Email: `vinodv@mit.edu`.

**Caltech. Email: `vidick@caltech.edu`.

††MIT. Email; `lisayang@mit.edu`.

Contents

1 Introduction

Efficient verification of computation is one of the most fundamental and intriguing concepts in computer science, and lies at the heart of the P vs. NP question. It has been studied in the classical setting for over three decades, giving rise to beautiful notions such as interactive proofs [GMR85], multi-prover interactive proofs [BGKW88], probabilistically checkable proofs [BFL90, ALM+92, AS92], and culminating with the notion of a *succinct* (interactive and non-interactive) *argument* [Kil92, Mic94]. Roughly speaking, a succinct argument for a T-time computation enables a prover running in $\mathsf{poly}(T)$ time to convince a $\mathsf{polylog}(T)$-time verifier of the correctness of the computation using only $\mathsf{polylog}(T)$ bits of communication, with soundness against all polynomial-time cheating provers.

In a breakthrough result in 2018, Mahadev [Mah18] presented an interactive argument system that enables a classical verifier to check the correctness of an arbitrary *quantum* computation. Mahadev's protocol represents a different kind of interactive argument — unlike the traditional setting in which the prover simply has more *computational resources* (i.e., running time) than the verifier, the prover in Mahadev's protocol works in a qualitatively more powerful *computational model.* More precisely, for any T-time quantum computation, Mahadev's protocol enables a quantum prover running in time $\mathsf{poly}(T)$ to convince a classical $\mathsf{poly}(T)$-time verifier with $\mathsf{poly}(T)$ bits of classical communication. Soundness holds against all quantum polynomial-time cheating provers under the post-quantum hardness of the learning with errors (LWE) problem.

A fundamental question is whether we can get the best of both worlds: can the prover have *both* a more powerful computational model *and* significantly greater computational resources? Namely, we want an interactive argument system for T-time quantum computation in which the quantum prover runs in $\mathsf{poly}(T)$ time and convinces a $\mathsf{polylog}(T)$-time classical verifier with $\mathsf{polylog}(T)$ bits of classical communication.

We answer this question affirmatively, both for $\mathsf{poly}(T)$-time quantum computations, corresponding to the complexity class **BQP**, and also for the non-deterministic analog **QMA**.

Theorem 1.1 (Succinct Arguments for **BQP**). *Let λ be a security parameter. Assuming the existence of a post-quantum secure indistinguishability obfuscation scheme* (iO) *and the post-quantum hardness of the learning with errors problem* (LWE)*, there is an interactive argument system for any T-time quantum computation on input x,*[1] *where*

- *the prover is quantum and runs in time* $\mathsf{poly}(T, \lambda)$*,*
- *the verifier is classical and runs in time* $\mathsf{poly}(\log T, \lambda) + \tilde{O}(|x|)$*,*[2] *and*
- *the protocol uses* $\mathsf{poly}(\log T, \lambda)$ *bits of classical communication.*

Theorem 1.2 (Succinct Arguments for **QMA**). *Assuming the existence of a post-quantum secure indistinguishability obfuscation scheme* (iO) *and the post-quantum hardness of the*

[1] A T-time quantum computation is a *language* L decidable by a bounded-error T-time quantum Turing machine [BV97]. We leave it to future work to address more complex tasks such as *sampling* problems (as in [CLLW20]).

[2] As in the classical setting, some dependence on $|x|$ is necessary at least to read the input; as in [Kil92], we achieve a fairly minimal $|x|$-dependence.

learning with errors problem (LWE)*, there is an interactive argument system for any T-time quantum computation on input x and a* $\mathsf{poly}(T)$*-qubit witness, where*

- *the prover is quantum and runs in time* $\mathsf{poly}(T, \lambda)$*, using polynomially many copies of the witness,*[3]
- *the verifier is classical and runs in time* $\mathsf{poly}(\log T, \lambda) + \tilde{O}(x)$*, and*
- *the protocol uses* $\mathsf{poly}(\log T, \lambda)$ *bits of classical communication.*

A New Proof of Security for the [Mah18] Protocol. One might hope to prove Theorems 1.1 and 1.2 by treating the Mahadev result as a "black box" and showing that *any* (classical) interactive argument for quantum computations can be compressed into a succinct protocol via a suitable cryptographic compiler. This is especially appealing given the extremely technical nature of Mahadev's security proof. Unfortunately, for reasons that will become clear in the technical overview, this kind of generic compilation seems unlikely to be achievable in our setting. Even worse, there does not appear to be any easily formalized property of the Mahadev protocol that would enable such a compilation.

Instead, our solution consists of two steps.

(1) We build a modified variant of the [Mah18] protocol and give an entirely self-contained proof of security. This modified protocol satisfies a few technical conditions that the original [Mah18] does not; most prominently, the *first verifier message* of our modified protocol is already succinct.

(2) We give a generic compiler that converts the protocol from Step (1) into a succinct argument system.

Our Step (1) also results in a self-contained proof of security of the original [Mah18] protocol that is more modular and amenable to further modification and generalization, which we believe will be useful for future work. Our analysis builds upon [Mah18] itself as well as an alternative approach described in Vidick's (unpublished) lecture notes [Vid20]. A concrete consequence of our new proof is that one of the two "hardcore bit" security requirements of the main building block primitive ("extended noisy trapdoor claw-free functions") in [Mah18] is not necessary.

Additional Results. Beyond our main result of succinct arguments for **BQP** and **QMA**, we explore a number of extensions and obtain various new protocols with additional properties.

- *Non-Interactive*: Although our protocols are not public-coin, we show how to modify them in order to apply the Fiat-Shamir transformation and round-collapse our protocols. As a result, we obtain designated-verifier non-interactive arguments for **BQP** (and the non-deterministic analog **QMA**) with security in the quantum random oracle model (QROM).

[3] We inherit the need for polynomially-many copies of the witness from prior works. This is a feature common to all previous classical verification protocols, and even to the quantum verification protocol of [FHM18].

- *Zero-Knowledge*: We show how to lift both variants of our protocol (interactive and non-interactive) to achieve zero-knowledge. We show a generic transformation based on classical two-party computation for reactive functionalities that makes our protocols simulatable. This transformation does not add any new computational assumption to the starting protocol.
- *Batch Arguments from LWE*: For the case of batch arguments, i.e., where the parties engage in the parallel verification of n statements, we show a succinct protocol that only assumes the post-quantum hardness of LWE (without iO). In this context, succinctness requires that the verifier's complexity scales with the size of a *single instance*, but is independent of n.

Prior Work. As discussed above, Mahadev [Mah18] constructs a *non-succinct* argument system for $\mathbf{BQP}/\mathbf{QMA}$ under LWE. The only prior work addressing *succinct* classical arguments for quantum computation is the recent work of Chia, Chung and Yamakawa [CCY20]. [CCY20] constructs a classically verifiable argument system for quantum computation in the following setting:

- The prover and verifier share a $\mathsf{poly}(T)$-bits long, structured reference string (which requires a trusted setup to instantiate) along with a hash function h (e.g. SHA-3).
- The "online communication" of the protocol is succinct ($\mathsf{poly}(\log T)$).
- Security is heuristic: it can be proved when h is modeled as a random oracle, but the *protocol description itself* explicitly requires the code of h (i.e. uses h in a non-black-box way).

We specifically note that when viewed in the *plain model* (i.e., without setup), the verifier must send the structured reference string to the prover, resulting in a protocol that is *not succinct*. We note that [CCY20] was specifically optimizing for a *two-message* protocol, but their approach seems incapable of achieving succinctness in the plain model even if further interaction is allowed.

By contrast, our succinct interactive arguments are in the plain model and are secure based on well-formed cryptographic assumptions, and our succinct 2-message arguments are proved secure in the QROM (and do not require a long common reference string).

Finally, we remark that our approach to achieving succinct arguments fundamentally (and likely necessarily) differs from [CCY20] because we manipulate the "inner workings" of the [Mah18] protocol; by contrast [CCY20] makes "black-box" use of a specific soundness property of the [Mah18] protocol (referred to as "computational orthogonality" by [ACGH20]) and is otherwise agnostic to how the protocol is constructed.

Acknowledgments. AL is supported in part by a Charles M. Vest fellowship. GM is partially supported by the German Federal Ministry of Education and Research BMBF (grant 16K15K042, project 6GEM). TV is supported by AFOSR YIP award number FA9550-16-1-0495, a grant from the Simons Foundation (828076, TV), MURI Grant FA9550-18-1-0161, the NSF QLCI program through grant number OMA-2016245 and the IQIM, an NSF Physics Frontiers Center (NSF Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028). AL, VV, and LY are supported in part by DARPA under Agreement No. HR00112020023, a grant from MIT-IBM Watson AI, a grant from Analog Devices, a Microsoft Trustworthy AI grant and the

Thornton Family Faculty Research Innovation Fellowship. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government or DARPA. LY was supported in part by an NSF graduate research fellowship.

2 Technical Overview

Our starting point is Mahadev's protocol for classical verification of quantum computation [Mah18], the core ingredient of which is a *measurement protocol*.

2.1 Recap: Mahadev's Measurement Protocol

We begin by reviewing Mahadev's N-qubit measurement protocol. In Mahadev's protocol, a quantum prover holding an N-qubit quantum state $\boldsymbol{\rho}$ interacts with a classical verifier, who wants to obtain the result of measuring $\boldsymbol{\rho}$ according to measurement bases $h \in \{0,1\}^N$ (h_i specifies a basis choice for the ith qubit, with $h_i = 1$ corresponding to the Hadamard basis and $h_i = 0$ corresponding to the standard basis).

Trapdoor Claw-Free Functions. At the heart of the protocol is a cryptographic primitive known as an *injective/claw-free trapdoor function* (a variant of lossy trapdoor functions [PW08, PVW08, GVW15]), which consists of two trapdoor function families Inj (for injective) and Cf (for claw-free), with the following syntactic requirements:[4]

- Each function in $\mathsf{Cf} \cup \mathsf{Inj}$ is indexed by a public-key pk, where functions $f_{\mathsf{pk}} \in \mathsf{Inj}$ are injective and functions $f_{\mathsf{pk}} \in \mathsf{Cf}$ are two-to-one. Moreover, pk can be sampled along with a secret key sk that enables computing f_{pk}^{-1} (i.e., $f_{\mathsf{pk}}^{-1}(y)$ consists of a single pre-image if $f_{\mathsf{pk}} \in \mathsf{Inj}$, and two pre-images if $f_{\mathsf{pk}} \in \mathsf{Cf}$).
- All functions in Inj and Cf have domain $\{0,1\}^{\ell+1}$ (for some ℓ) and the two pre-images of y under $f_{\mathsf{pk}} \in \mathsf{Cf}$ are of the form $(0, x_0)$ and $(1, x_1)$ for some $x_0, x_1 \in \{0,1\}^{\ell}$.

An injective/claw-free trapdoor function must satisfy the following security properties:[5]

1. **Claw-Free/Injective Indistinguishability.** A random function in $f_{\mathsf{pk}} \leftarrow \mathsf{Cf}$ is computationally indistinguishable from a random function $f_{\mathsf{pk}} \leftarrow \mathsf{Inj}$.
2. **Adaptive Hardcore Bit.** Given $f_{\mathsf{pk}} \leftarrow \mathsf{Cf}$, it is computationally infeasible to output both (1) a pair (x, y) satisfying $f_{\mathsf{pk}}(x) = y$ and (2) a non-zero string $d \in \{0,1\}^{\ell+1}$ such that $d \cdot (1, x_0 \oplus x_1) = 0$, where $(0, x_0)$ and $(1, x_1)$ are the two preimages of y.[6]

[4] The actual syntactic requirements, described in Section 3.5, are somewhat more complex due to the fact that the functions in question are probabilistic.

[5] In fact, Mahadev's proof relies on two different hardcore bit properties, but we show in this work that only the adaptive hardcore bit property is needed.

[6] The full definition places a slightly stronger restriction on d than simply being non-zero. However, this simplified version will suffice for this overview.

To build some intuition about the usefulness of such function families, notice that they can be used to commit to a single classical bit quite easily. The commitment key is a function $f_{\mathsf{pk}} \in \mathsf{Inj}$, and commitment to a bit b is $y = f_{\mathsf{pk}}(b, x)$ for a random $x \in \{0,1\}^n$. It is not hard to verify that this is a statistically binding and computationally hiding commitment of b. On the other hand, if $f_{\mathsf{pk}} \in \mathsf{Cf}$, it is a statistically hiding and computationally binding commitment of b.[7]

Protocol Description. With this intuition in mind, we now describe a (slightly simplified version of) the Mahadev measurement protocol. Mahadev's protocol is a many-fold sequential repetition of the following one-bit challenge protocol.

$\mathbf{V} \to \mathbf{P}$**:** The verifier samples N pairs of functions and their trapdoors: for every $i \in [\ell]$, he samples $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Inj}$ if $h_i = 0$ and $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Cf}$ if $h_i = 1$. He sends the N-tuple of public keys (function descriptions) $\underline{(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)}$ to the prover.

$\mathbf{P} \to \mathbf{V}$**:** The prover holds an N-qubit state $\boldsymbol{\rho}$ on register $\mathcal{B} = \mathcal{B}_1 \otimes \cdots \mathcal{B}_N$. She initializes registers $\mathcal{X} = \mathcal{X}_1 \otimes \cdots \mathcal{X}_N$ so that each $\mathcal{X}_i$ is an ℓ-qubit register in uniform superposition. Then she evaluates each f_{pk_i} on the $\ell + 1$ qubit register $\mathcal{B}_i \otimes \mathcal{X}_i$ and measures the output to obtain y_i. Finally, she sends $\underline{(y_1, \ldots, y_N)}$ to the verifier, keeping the registers $\mathcal{B}, \mathcal{X}$ for herself.

$\mathbf{V} \to \mathbf{P}$**:** The verifier picks a random challenge bit $\underline{c \leftarrow \{0,1\}}$ and sends it to the prover. $c = 0$ indicates a "test round" and $c = 1$ indicates a "measurement round".

$\mathbf{P} \to \mathbf{V}$**:** If $c = 0$ ("test"), the prover measures each $\mathcal{B}_i \otimes \mathcal{X}_i$ in the standard basis, obtaining $(b_i, x_i) \in \{0,1\} \times \{0,1\}^\ell$, and sends $\underline{(b_i, x_i)_{i=1}^N}$ to the verifier.

If $c = 1$ ("measure"), she measures each $\mathcal{B}_i \otimes \mathcal{X}_i$ in the Hadamard basis (i.e., she applies the Hadamard gate H on each qubit and then measures in the standard basis), obtaining $d_i \in \{0,1\}^{\ell+1}$, and sends $\underline{(d_i)_{i=1}^N}$ to the verifier.

$\mathbf{V}$ **(output):** If $c = 0$ ("test"), $\mathbf{V}$ either accepts or rejects the interaction but outputs nothing. In particular, $\mathbf{V}$ checks that $f_{\mathsf{pk}_i}(b_i, x_i) = y_i$. If any of the checks fail, reject.

If $c = 1$ ("measure"), proceed as follows. Let $R \subseteq [N]$ be the set of indices where $h_i = 1$, corresponding to Hadamard basis measurements, and let $S = [N] \setminus R$ be the set of indices corresponding to standard basis measurements.

For each $i \in R$, $\mathbf{V}$ computes the two inverses $(0, x_{i,0})$ and $(1, x_{i,1})$ of y_i (which are guaranteed to exist since $f_{\mathsf{pk},i} \in \mathsf{Cf}$) using sk_i. $\mathbf{V}$ sets $u_i := d \cdot (1, x_{i,0} \oplus x_{i,1})$ as the i^{th} measurement outcome. For every $i \in S$, $\mathbf{V}$ ignores d_i, and sets v_i to be the first bit of $f_{\mathsf{pk}_i}^{-1}(y_i)$, computed using the trapdoor sk_i (this is well-defined since $f_{\mathsf{pk},i} \in \mathsf{Inj}$). Finally $\mathbf{V}$ outputs the N-bit string $\underline{(u, v) \in \{0,1\}^R \times \{0,1\}^S}$.

Mahadev [Mah18] proves that if a malicious prover $\mathbf{P}^*$ passes the test round with probability 1, then there exists an N-qubit quantum state $\boldsymbol{\rho}^*$ — *independent* of the verifier's measurement basis h — such that the result of measuring $\boldsymbol{\rho}^*$ according to h is computationally indistinguishable from

[7] In particular, $f_{\mathsf{pk}} \in \mathsf{Cf}$ satisfies Unruh's definition of *collapse-binding* [Unr16b].

the verifier's N-bit output distribution in the measurement round.[8] While her definition requires that such a $\boldsymbol{\rho}^*$ *exists*, Vidick and Zhang [VZ21] showed that Mahadev's proof steps implicitly define an extractor that efficiently produces $\boldsymbol{\rho}^*$ using black-box access to $\mathbf{P}^*$.

2.2 Defining a (Succinct) Measurement Protocol

Our first (straightforward but helpful) step is to give an explicit definition of a *commit-and-measure protocol* that abstracts the completeness and soundness properties of Mahadev's measurement protocol as established in [Mah18, VZ21]. Roughly speaking, a commit-and-measure protocol is sound if, for any malicious prover $\mathbf{P}^*$ that passes the test round with probability 1 and any basis choice h, there exists an efficient extractor that (without knowledge of h) interacts with prover and outputs an extracted state $\boldsymbol{\tau}$ such that the following are indistinguishable:

- the distribution of verifier outputs obtained in the measurement round from interacting with $\mathbf{P}^*$ using basis choice h, and
- the distribution of measurement outcomes obtained from measuring $\boldsymbol{\tau}$ according to h.

This abstraction will be particularly helpful for reasoning about our eventual *succinct* measurement protocols, which will necessitate modifying Mahadev's original protocol.

Can a Measurement Protocol be Succinct? Given the definition of a measurement protocol, an immediate concern arises with respect to obtaining succinct arguments: the verifier's *input* to the measurement protocol – the basis vector h – is inherently non-succinct. Since the number of qubits N grows with the runtime of the BQP computation when used to obtain quantum verification [FHM18], this poses an immediate problem.

Our solution to this problem is to only consider basis vectors h that are *succinct*; our formalization is that h must be the truth table of an efficiently computable function $f : [\log N] \to \{0, 1\}$. For any such h, we can represent the verifier's input as a circuit C that computes h, removing the above obstacle.

However, in order for there to be any hope of this idea working, it must be the case that measurement protocols for bases with succinct representations are still useful for constructing delegation for BQP. Fortunately, it has been shown [ACGH20] that classically verifiable (non-succinct) arguments for BQP can be constructed by invoking Mahadev's measurement protocol (and, by inspection of the proof, any measurement protocol satisfying our definition) on a *uniformly random basis string* $h \leftarrow \{0, 1\}^N$. Then, by computational indistinguishability, it is also possible to use a *pseudorandom* string h that has a succinct representation, i.e., $h = (\mathsf{PRF}_s(1), \ldots, \mathsf{PRF}_s(N))$ for some (post-quantum) pseudorandom function PRF.

Thus, we focus for the moment on constructing a succinct measurement protocol for h with succinct representation, and return to the full delegation problem later.

[8] This can be extended to provers that pass the test round with probability $1 - \varepsilon$ by the gentle measurement lemma. In particular, an efficient distinguisher can only distinguish the verifier's output distribution from the result of measuring some $\boldsymbol{\rho}^*$ with advantage $\mathsf{poly}(\varepsilon)$.

2.3 Constructing a Verifier-Succinct Measurement Protocol

Inspecting the description of the [Mah18] protocol, there are three distinct reasons that the protocol is not succinct:

1. The verifier's first message, which consists of N TCF public keys, is non-succinct.
2. The prover's two messages, consisting of the commitments y_i and openings z_i respectively, are non-succinct.
3. The verifier's decision predicate, as it is a function of these commitments and openings, requires $\mathsf{poly}(N)$ time to evaluate.

The latter two issues turn out to be not too difficult to resolve (although there is an important subtlety that we discuss later); for now, we focus on resolving (1), which is our main technical contribution. Concretely, we want to construct a measurement protocol for succinct bases h where the verifier's first message is succinct.

Idea: Compress the Verifier's message with iO. Given the problem formulation, a natural idea presents itself: instead of having V send over N i.i.d. public keys pk_i, perhaps V can send a succinct program PK that contains the description of N public keys pk_i that are in some sense "pseudoindependent!" Using the machinery of obfuscation and the "punctured programs" technique [SW14], it is straightforward to write down a candidate program for this task: simply obfuscate the following code.

```
Input: index $i \leq N$
Hardwired Values: Puncturable PRF seed $s$. Circuit $C$.

  • Compute $\mathsf{mode} = C(i)$ and $r = \mathsf{PRF}_s(i)$.
  • Compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode}; r)$.
  • Output $\mathsf{pk}_i$.
```

Here, C is an efficient circuit with truth table h, and $\mathsf{Gen}(1^\lambda, \mathsf{mode})$ indicates sampling either from Inj or Cf depending on whether $h_i = C(i) = 0$ or $h_i = C(i) = 1$.

Letting PK denote an obfuscation of the above program, V could send PK to P and allow the prover to compute each $\mathsf{pk}_i = \mathsf{PK}(i)$ on its own, and the protocol could essentially proceed as before, except that the verifier will have to expand its PRF seed s into $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$ in order to compute its final output.

Problem: Proving Soundness. While it is not hard to describe this plausible modification to the [Mah18] protocol that compresses the verifier's message, it is very unclear how to argue that the modified protocol is sound. The obfuscation literature has no shortage of proof techniques developed over the last 10 years, but since we have made a "non-black-box" modification of the [Mah18] protocol, a deep understanding of the [Mah18] proof of soundness is required in order to understand to what extent these techniques are compatible with the application at hand.

We believe it *should* be possible to incorporate punctured programming techniques into Mahadev's proof of soundness in [Mah18] and conclude the desired soundness property of the new protocol. However, doing so would result in an extremely complex proof that would require the reader to verify the entirety of the [Mah18] (already very complicated) original security proof with our modifications in mind.

2.4 Proof of Soundness

Given the complicated nature of the [Mah18] proof of soundness, we instead give a *simpler* and *more modular* proof of soundness for the [Mah18] measurement protocol. Moreover, we give this proof for a generic variant of the [Mah18] protocol where the prover is given an arbitrary representation PK of N TCF public keys and show that precisely two properties of this representation PK are required in order for the proof to go through:

- An appropriate generalization of the "dual-mode" property of individual TCFs must hold for PK: for any two circuits C_1, C_2, it should be that PK_1 generated from basis C_1 is computationally indistinguishable from PK_2 generated from basis C_2. In fact, a stronger variant of this indistinguishability must hold: it should be the case that $\mathsf{PK}_1 \approx_c \mathsf{PK}_2$ even if the distinguisher is given all secret keys sk_j such that $C_1(j) = C_2(j)$.
- For every i, the adaptive hardcore bit property of f_{pk_i} should hold *even given* sk_j for all $j \neq i$.

Since these two properties are (essentially) all that is required for our proof to go through, in order to obtain a verifier-succinct protocol, it suffices to show that the obfuscated program PK above satisfies these two properties, which follows from standard techniques.

Thus, we proceed by describing our new soundness proof for the [Mah18] measurement protocol, which transparently generalizes to the verifier-succinct setting.

The "Operational Qubits" Approach. Let P^* denote a prover that passes the test round (i.e., makes the verifier accept on the 0 challenge) with probability 1. Our goal is to show that the prover in some sense "has an N-qubit state" such that measuring this state in the h-bases produces the same (or an indistinguishable) distribution as the verifier's protocol output, which we will denote $D_{P^*,\mathrm{Out}}$. This N-qubit state should be efficiently computable from the prover's internal state $|\psi\rangle$; specifically, we use $|\psi\rangle$ to denote the prover's state after its first message y has been sent.

In order to show this, taking inspiration from [Vid20],[9] we will proceed in two steps:

[9] [Vid20] gives a soundness proof for a variant of the [Mah18] protocol, but in a qualitatively weaker setting.

1. Identify N "operational qubits" within $|\psi\rangle$. That is, we will identify a set of $2N$ observables $Z_1, \ldots, Z_N, X_1, \ldots, X_N$ (analogous to the "Pauli observables" $\sigma_{z,1}, \ldots, \sigma_{z,N}, \sigma_{x,1}, \ldots \sigma_{x,N}$) such that measuring $|\psi\rangle$ with these observables gives the outcome distribution $D_{P^*,\mathrm{Out}}$.

   Provided that these $2N$ observables roughly "behave like" Pauli observables with respect to $|\psi\rangle$ (e.g. satisfy the X/Z uncertainty principle), one could then hope to:

2. Extract a related state $|\psi'\rangle$ such that measuring $|\psi'\rangle$ in the *actual* standard/Hadamard bases matches the "pseudo-Pauli" $\{Z_j\}$, $\{X_i\}$, measurements of $|\psi\rangle$ (and therefore $D_{P^*,\mathrm{Out}}$).

Relating the Verifier's Output to Measuring $|\psi\rangle$. Our current goal is to achieve Step (1) above. Let $|\psi\rangle$ denote P^*'s post-commitment state and let U denote the unitary such that P^*'s opening is a measurement of $U|\psi\rangle$ in the Hadamard basis.

Now, let us consider the verifier's output distribution. The ith bit of the verifier's output when $h_i = 1$ is defined to be $d \cdot (x_{0,i} \oplus x_{1,i})$ (where d is the opening sent by the prover) of $U|\psi\rangle$ in the Hadamard basis. For each such i, we can define an observable X_i characterizing this measurement, that *roughly* takes the form

$$X_i \approx U^\dagger (H_{\mathcal{Z}_i} \otimes \mathsf{Id}) \left(\sum_d (-1)^{d\cdot(1,x_{0,i}\oplus x_{1,i})} |d\rangle\langle d|_{\mathcal{Z}_i} \otimes \mathsf{Id}_{\mathcal{I},\{\mathcal{Z}_j\}_{j\neq i}} \right) (H_{\mathcal{Z}_i} \otimes \mathsf{Id}) U.$$

Here we have slightly simplified the expression for X_i for the sake of presentation; the correct definition of X_i (see Section 6.2) must account for the case where d is rejected by the verifier. To reiterate, the observable X_i is a syntactic interpretation of the verifier's output m_i as a function of $|\psi\rangle$.

On the other hand, when $h_i = 0$, the verifier's output m_i is not *a priori* a measurement of $|\psi\rangle$; indeed, the verifier ignores the prover's second message and just inverts y_i. However, under the assumption that the prover P^* passes the test round with probability $1 - \mathsf{negl}(\lambda)$, making use of the fact that f_{pk_i} is injective, this y_i-inverse must be equal to what the prover *would* have sent in the test round. This defines another observable on $|\psi\rangle$ that we call Z_i:

$$Z_i = \sum_{b,x} (-1)^b |b,x\rangle\langle b,x|_{\mathcal{Z}_i} \otimes \mathsf{Id}_{\mathcal{I},\{\mathcal{Z}_j\}_{j\neq i}}.$$

Finally, note that the operator Z_i syntactically makes sense even when $h_i = 1$. However, X_i cannot even be *defined* when f_{pk_i} is injective, corresponding to $h_i = 0$, since X_i explicitly requires *two* inverses of y_i. Therefore, from now on, we sample all $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Cf}$ (forcing all TCFs to be 2-to-1).

This brings us to the punchline of this step: by invoking a computational assumption (the indistinguishability of Cf and Inj), we can define observables (X_i, Z_i) for all $i \in [N]$ such that for *every* i and *every* basis choice h, the distribution resulting from measuring $|\psi\rangle$ with X_i (resp. Z_i) matches the ith bit of the verifier's output distribution.

[Vid20] only proves indistinguishability of N-qubit measurements that are either *all* in the standard basis or *all* in the Hadamard basis, and only proves indistinguishability with respect to *linear* tests of the distribution (that is, [Vid20] proves small-bias rather than full indistinguishability). Both of these relaxations are unacceptable in our setting, and achieving the latter specifically requires a different proof strategy.

With a little more work, one can actually show that the verifier's *entire* output distribution in the h-basis is computationally indistinguishable from the following distribution $D_{P^*,\textsf{2-to-1}}$:

- Sample keys $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Cf}$. Run P^* to obtain $y, |\psi\rangle$.
- For each i such that $h_i = 0$, measure the first bit of the prover's ith response register in the standard basis to obtain (and output) a bit b_i.
- Measure $U|\psi\rangle$ in the Hadamard basis, obtaining strings $(d_1, \ldots, d_N)$.
- For each i such that $h_i = 1$, compute (and output) $d_i \cdot (1, x_{0,i} \oplus x_{1,i})$.

Aside: Why are these Z_j and X_i helpful? As alluded to earlier, this approach is inspired by *operational* definitions of "having an N-qubit state," which consists of a state $|\psi\rangle$ and $2N$ "pseudo-Pauli" observables $Z_1, \ldots, Z_N, X_1, \ldots X_N$ that behave "like Pauli observables" on $|\psi\rangle$. For example, it is possible to prove that many of the "Pauli group relations" hold *approximately* on these X_i, Z_j with respect to $|\psi\rangle$, meaning that (for example)

$$\langle\psi| Z_i X_i Z_i + X_i |\psi\rangle = \mathsf{negl}(\lambda)$$

and

$$\langle\psi| Z_j X_i Z_j - X_i |\psi\rangle = \mathsf{negl}(\lambda)$$

for $i \neq j$. In fact, these relations turn out to *encode* the two basic properties of the TCF f_{pk_i}: the adaptive hardcore bit property (encoded in the first relation) and that f_{pk_i} is indistinguishable from injective[10] (encoded in the second relation)! We will not directly prove the relations here, but they are implicit in our full security proof and are the motivation for this proof strategy.

The Extracted State. Given these protocol observables $Z_1, \ldots, Z_N, X_1, \ldots, X_N$, it remains to implement Step (2) of our overall proof strategy: extracting a state $|\psi'\rangle$ whose standard/Hadamard measurement outcomes match $D_{P^*,\mathrm{Out}}$. At a high level, this is achieved by "teleporting" the state $|\psi\rangle$ onto a fresh N-qubit register in a way that *transforms* the "pseudo-Paulis" $\{X_i\}, \{Z_j\}$ into *real* Pauli observables $\{\sigma_{x,i}\}, \{\sigma_{z,j}\}$.

Fix a choice of $\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp}$. For ease of notation, write $\mathcal{H} = \mathcal{Z} \otimes \mathcal{I} \otimes \mathcal{U}$ so that $|\psi\rangle \in \mathcal{H}$. We would like an efficient extraction procedure that takes as input $|\psi\rangle \in \mathcal{H}$ and generates an N-qubit state $\boldsymbol{\tau}$ such that, roughly speaking, measuring $|\psi\rangle$ with X/Z and measuring $\boldsymbol{\tau}$ with σ_X/σ_Z produce indistinguishable outcomes.

Intuition for the Extractor. Before we describe our extractor, we first provide some underlying intuition. For an arbitrary N-qubit Hilbert space, let $\sigma_{x,i}/\sigma_{z,i}$ denote the Pauli σ_x/σ_z observable acting on the ith qubit. For each $r, s \in \{0,1\}^N$, define the N-qubit Pauli "parity" observables

$$\sigma_x(r) := \prod_{i:r_i=1} \sigma_{x,i} \ , \ \sigma_z(s) := \prod_{i:r_i=1} \sigma_{z,i}.$$

[10] Technically, the property encoded is the *collapsing* of f_{pk_i}, which is implied by (but not equivalent to) being indistinguishable from injective.

Suppose for a moment that $|\psi\rangle \in \mathcal{H}$ is *already* an N-qubit state (i.e., $\mathcal{H}$ is an N-qubit Hilbert space) and moreover, that each X_i/Z_i observable is simply the corresponding Pauli observable $\sigma_{x,i}/\sigma_{z,i}$. While these assumptions technically trivialize the task (the state already has the form we want from the extracted state), it will be instructive to **write down an extractor that "teleports" this state into another N-qubit external register.**

We can do this by initializing two N-qubit registers $\mathcal{A}_1 \otimes \mathcal{A}_2$ to $|\phi^+\rangle^{\otimes N}$ where $|\phi^+\rangle$ is the EPR state $(|00\rangle + |11\rangle)/\sqrt{2}$ (the ith EPR pair lives on the ith qubit of $\mathcal{A}_1$ and $\mathcal{A}_2$). Now consider the following steps, which are inspired by the (N-qubit) quantum teleportation protocol

1. Initialize a $2N$-qubit ancilla $\mathcal{W}$ to $\left|0^{2N}\right\rangle$, and apply $H^{\otimes 2N}$ to obtain the uniform superposition.
2. Apply a "controlled-Pauli" unitary, which does the following for all $r, s \in \{0,1\}^N$ and all $|\phi\rangle \in \mathcal{H} \otimes \mathcal{A}_1$:

$$|r,s\rangle_{\mathcal{W}} |\phi\rangle_{\mathcal{H},\mathcal{A}_1} \to |r,s\rangle_{\mathcal{W}} \left(\sigma_x(r)\sigma_z(s)_{\mathcal{H}} \otimes \sigma_x(r)\sigma_z(s)_{\mathcal{A}_1}\right) |\phi\rangle_{\mathcal{H},\mathcal{A}_1}$$

3. Apply the unitary that XORs onto $\mathcal{W}$ the outcome of performing N Bell-basis measurements[11] on $\mathcal{A}_1 \otimes \mathcal{A}_2$ onto $\mathcal{W}$, i.e., for all $u, v, r, s \in \{0,1\}^N$:

$$|u,v\rangle_{\mathcal{W}} (\sigma_x(r)\sigma_z(s) \otimes \mathsf{Id})_{\mathcal{A}_1,\mathcal{A}_2} \left|\phi^+\right\rangle^{\otimes N}_{\mathcal{A}_1,\mathcal{A}_2} \to |u \oplus r, v \oplus s\rangle_{\mathcal{W}} (\sigma_x(r)\sigma_z(s) \otimes \mathsf{Id})_{\mathcal{A}_1,\mathcal{A}_2} \left|\phi^+\right\rangle^{\otimes N}_{\mathcal{A}_1,\mathcal{A}_2}.$$

   Finally, discard $\mathcal{W}$.

One can show that the resulting state is

$$\frac{1}{2^N} \sum_{r,s \in \{0,1\}^N} (\sigma_x(r)\sigma_z(s) \otimes \sigma_x(r)\sigma_z(s) \otimes \mathsf{Id}) |\psi\rangle_{\mathcal{H}} \left|\phi^+\right\rangle_{\mathcal{A}_1,\mathcal{A}_2} = \left|\phi^+\right\rangle_{\mathcal{H},\mathcal{A}_1} |\psi\rangle_{\mathcal{A}_2}, \tag{1}$$

where $|\psi\rangle$ is now "teleported" into the $\mathcal{A}_2$ register.

The Full Extractor. To generalize this idea to the setting where $|\psi\rangle \in \mathcal{H}$ is an arbitrary quantum state and $\{X_i, Z_i\}_i$ are an arbitrary collection of $2N$ observables, *we simply replace each $\sigma_x(r)$ and $\sigma_z(s)$ acting on $\mathcal{H}$ above with the corresponding parity observables $X(r)$, $Z(s)$*, defined analogously (for $r, s \in \{0,1\}^N$ as

$$Z(s) = \prod_{i=1}^{N} Z_i^{s_i} \quad \text{and} \quad X(r) = \prod_{i=1}^{N} X_i^{r_i}.$$

The rough intuition is that as long as the $\{X_i\}$ and $\{Z_i\}$ observables "behave like" Pauli observables with respect to $|\psi\rangle$, the resulting procedure will "teleport" $|\psi\rangle$ into the N-qubit register $\mathcal{A}_2$.

[11] The Bell basis consists of the 4 states $(\sigma_x^a \sigma_z^b \otimes \mathsf{Id}) \left|\phi^+\right\rangle$ for $a, b \in \{0,1\}$ on 2 qubits.

Relating Extracted State Measurements to Verifier Outputs. With the extracted state defined to be the state on $\mathcal{A}_2$ after performing the "generalized teleportation" described above, it remains to prove that the distribution $D_{P^*,\mathrm{Ext}}$ resulting from measuring the extracted state on $\mathcal{A}_2$ in the h-bases is indistinguishable from $D_{P^*,\text{2-to-1}}$.

One can show (by a calculation) that $D_{P^*,\mathrm{Ext}}$ is the following distribution (differences from $D_{P^*,\text{2-to-1}}$ in red)

1. Sample keys $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Cf}$. Run P^* to obtain $y, |\psi\rangle$.
2. For each i such that $h_i = 0$, measure the first bit of the prover's ith response register in the standard basis to obtain (and output) a bit b_i.
3. For each i such that $h_i = 1$, flip a random bit w_i and apply the unitary $Z_i^{w_i}$.
4. Measure $U|\psi\rangle$ in the Hadamard basis, obtaining strings $(d_1, \ldots, d_N)$.
5. For each i such that $h_i = 1$, compute (and output) $d_i \cdot (1, x_{0,i} \oplus x_{1,i}) \oplus w_i$.

We prove indistinguishability between the N-bit distributions $D_{P^*,\mathrm{Ext}}$ and $D_{P^*,\text{2-to-1}}$ by considering N hybrid distributions, where the difference between Hybrid $j-1$ and Hybrid j is:

- an additional application of the unitary Z_j in Item 3, and
- an additional XOR of e_j (the jth standard basis vector) in Item 5.

To conclude the soundness proof, we show that Hybrid $j-1$ and Hybrid j in the following three steps.

- First, we prove that the marginal distributions of Hybrid $(j-1)$ and Hybrid j on $N \setminus \{j\}$ are indistinguishable due to the collapsing property of f_{pk_j}. Intuitively this holds because the marginal distributions on $N \setminus \{j\}$ only differ by the application of Z_j, which is undetectable by collapsing.
- By invoking an elementary lemma about N-bit indistinguishability, the task reduces to proving a 1-bit indistinguishability of the jth bit of Hybrid $(j-1)$ and Hybrid j, conditioned on an efficiently computable property of the marginal distributions on $N \setminus \{j\}$.
- Finally, we show that the indistinguishability of the jth bit holds due to the adaptive hardcore bit property of f_{pk_j}. At a very high level, the above jth bit property involves a measurement of X_j, and the two hybrids differ in whether a random Z_j^b is applied before X_j is measured; in words, this exactly captures the adaptive hardcore bit security game.

  We refer the reader to Section 6.4 for a full proof of indistinguishability.

2.5 From a Verifier-Succinct Measurement Protocol to Succinct Arguments for BQP

Using Sections 2.3 and 2.4, we have constructed a *verifier-succinct* measurement protocol, for succinctly represented basis strings, with a single bit verifier challenge. What remains is to convert this into a (fully) succinct argument system for BQP (or QMA). This is accomplished via the following transformations:

- Converting a measurement protocol into a quantum verification protocol. As described earlier, this is achieved by combining the [FHM18] protocol for BQP verification with a limited quantum verifier (as modified by [ACGH20]) with our measurement protocol, using a PRF to generate a pseudorandom basis choice instead of a uniformly random basis choice for the [FHM18, ACGH20] verifier. This results in a verifier-succinct argument system for BQP/QMA with constant soundness error.

- Parallel repetition to reduce the soundness error. This follows from the "computational orthogonal projectors" property of the 1-bit challenge protocol and follows from [ACGH20] (we give a somewhat more abstract formulation of their idea in Appendix A). This results in a verifier-succinct argument system for BQP/QMA with negligible soundness error.

- Converting a verifier-succinct argument system into a fully succinct argument system. We elaborate on this last transformation below, as a few difficulties come up in this step.

Assume that we are given a (for simplicity, 4-message) verifier-succinct argument system for BQP/QMA. Let m_1, m_2, m_3, m_4 denote the four messages in such an argument system. In order to obtain a fully succinct argument system, we must reduce (1) the prover communication complexity $|m_2| + |m_4|$, and (2) the runtime of the verifier's decision predicate.

The first idea that comes to mind is to ask the prover to send short (e.g. Merkle tree) commitments σ_2 and σ_4 of m_2 and m_4, respectively, instead of sending m_2 and m_4 directly. At the end of the interaction, the prover and verifier could then engage in a succinct interactive argument (of knowledge) for a (classical) NP statement that "the verifier would have accepted the committed messages underlying σ_2 and σ_4". One could potentially employ Kilian's succinct interactive argument of knowledge for NP which was recently shown to be post-quantum secure under the post-quantum LWE assumption [CMSZ21].

There are a few issues with this naive idea. First of all, the verifier's decision predicate is *private* (it depends on the secret key SK in the measurement protocol and the PRF seed for its basis), so the NP statement above is not well-formed. One reasonable solution to this issue is to simply have the verifier send this secret information st after the verifier-succinct protocol emulation has occurred and before the NP-succinct argument has started. For certain applications (e.g. obtaining a non-interactive protocol in the QROM) we would like to have a *public-coin* protocol; this can be achieved by using fully homomorphic encryption to encrypt this secret information in the *first* round rather than sending it in the clear in a later round. For this overview, we focus on the private-coin variant of the protocol.

Now, we can indeed write down the appropriate NP relation[12]

$$\begin{aligned}\mathcal{R}_V = &\{((h, m_1, \sigma_2, m_3, \sigma_4, \mathsf{st}), (m_2, m_4)) : \sigma_2 = h(m_2) \text{ and} \\ &\sigma_4 = h(m_4) \text{ and } V(\mathsf{st}, m_1, m_2, c, m_4) = \mathsf{accept}\}\end{aligned}$$

and execute the aforementioned strategy. However, this construction turns out not to work. Specifically, it does not seem possible to *convert* a cheating prover P^* in the above fully succinct protocol into a cheating prover P^{**} for the verifier-succinct protocol; for example, P^{**} needs to be able to produce a message m_2 given only m_1 from the verifier; meanwhile, the message m_1 can only be extracted from P^* by repeatedly rewinding P^*'s *last* message algorithm, which requires the verifier's secret information st as input! This does not correspond to a valid P^{**}, who does not have access to st when computing m_2.

Our refined compiler is to execute several arguments of knowledge: one right after the prover sends σ_2, proving knowledge of m_2; another one right after she sends σ_4, proving knowledge of m_4 (both before receiving the secret state st from the verifier); and a third one for the relation $\mathcal{R}_V$ described above. The first two arguments of knowledge are for the relation

$$\mathcal{R}_H = \{(h, \sigma), m) : h(m) = \sigma\}$$

This allows for *immediate extraction* of m_2 and m_3 and appears to clear the way for a reduction between the verifier-succinct and fully succinct protocol soundness properties.

However, there is one remaining problem: the argument-of-knowledge property of Kilian's protocol proved by [CMSZ21] is *insufficiently composable* to be used in our compiler. They demonstrate an extractor for Kilian's protocol that takes any quantum cheating prover that convinces the verifier and extracts a witness from them. However, their post-quantum extractor might significantly disturb the prover's state, meaning that once we extract m_2 above, we may not be able to continue the prover execution in our reduction.

Fortunately, a recent work [LMS21] shows that a slight variant of Kilian's protocol is a succinct argument of knowledge for NP satisfying a composable extraction property called "state-preservation." This security property is exactly what is required for our compiler to extract a valid cheating prover strategy P^{**} for the verifier-succinct argument given a cheating prover P^* for the compiled protocol. A full discussion of this is given in Section 9.

This completes our construction of a succinct argument system for BQP (and QMA). We discuss additional results (2-message protocols, zero knowledge, batch arguments) in Section 10.

3 Preliminaries

3.1 Quantum Information

Let $\mathcal{H}$ be a finite-dimensional Hilbert space. A pure state is a unit vector $|\psi\rangle \in \mathcal{H}$. Let $\mathrm{D}(\mathcal{H})$ denote the set of all positive semidefinite operators on $\mathcal{H}$ with trace 1. A mixed state is an operator $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{H})$, and is often called a *density matrix*. We sometimes divide $\mathcal{H}$ into named *registers* written in uppercase calligraphic font, e.g., $\mathcal{H} = \mathcal{A} \otimes \mathcal{B} \otimes \mathcal{C}$.

[12] Note that the verifier also takes as input the QMA instance, but we suppress it here for clarity.

For a density matrix $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{H})$, where $\mathcal{H} \simeq (\mathbb{C}^2)^{\otimes \ell}$, we sometimes use the shortcut $M(h, \boldsymbol{\rho})$ to denote the distribution resulting from measuring each qubit of $\boldsymbol{\rho}$ (where the qubits are specified by the isomorphism $\mathcal{H} \simeq (\mathbb{C}^2)^{\otimes \ell}$) in the basis determined by $h \in \{0,1\}^\ell$. By convention, $h_i = 0$ corresponds to measuring the i-th register in the standard basis $\{|0\rangle, |1\rangle\}$ and $h_i = 1$ corresponds to measuring the i-th register in the Hadamard basis $\{|+\rangle, |-\rangle\}$.

An *observable* is represented by a Hermitian operator O on $\mathcal{H}$. In particular, any observable O can be written in the form $\sum_i \lambda_i \Pi_i$ where $\{\lambda_i\}$ are real numbers and $\sum_i \Pi_i = \mathsf{Id}$. The measurement corresponding to an observable O is the projective measurement $\{\Pi_i\}$ with corresponding outcomes $\{\lambda_i\}$. A *binary observable* satisfies the additional requirement that $O^2 = \mathsf{Id}$. Notice that for any binary observable, O is a unitary matrix with eigenvalues in $\{1, -1\}$. In this case we sometimes treat the outcomes as bits through the usual correspondence $1 \to 0$, $-1 \to 1$.

Given a binary observable O, we define its corresponding *projection operators* $O^+ = \frac{1}{2}(\mathsf{Id} + O)$ and $O^- = \frac{1}{2}(\mathsf{Id} - O)$. O^+ and O^- correspond to projecting onto the $+1$ and -1 eigenspaces of O, respectively, and thus form a binary projective measurement.

The Class QMA. A language $\mathcal{L} = (\mathcal{L}_{\mathsf{yes}}, \mathcal{L}_{\mathsf{no}})$ is in **QMA** if and only if there is a uniformly generated family of polynomial-size quantum circuits $\mathcal{V} = \{V_\lambda\}_{\lambda \in \mathbb{N}}$ such that for every λ, V_λ takes as input a string $x \in \{0,1\}^\lambda$ and a quantum state $|\phi\rangle$ on $p(\lambda)$ qubits and returns a single bit and moreover the following conditions hold.

- For all $x \in \mathcal{L}_{\mathsf{yes}}$ of length λ, there exists a quantum state $|\psi\rangle$ on at most $p(\lambda)$ qubits such that the probability that V_λ accepts $(x, |\phi\rangle)$ is at least $2/3$. We denote the (possibly infinite) set of quantum states (which we will also refer to as quantum witnesses) that make V_λ accept x by $\mathcal{R}(x)$.

- For all $x \in \mathcal{L}_{\mathsf{no}}$ of length λ, and all quantum states $|\psi\rangle$ on at most $p(\lambda)$ qubits, it holds that V_λ accepts on input $(x, |\psi\rangle)$ with probability at most $1/3$.

3.2 Black-Box Access to Quantum Algorithms

Let A be a polynomial-time quantum algorithm with internal state $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{I})$ that takes a classical input r and produces a classical output z. Without loss of generality, the behavior of A can be described as follows:

1. Apply an efficient *classical* algorithm to r to generate the description of a unitary $U(r)$.

2. Initialize registers $\mathcal{Z} \otimes \mathcal{I}$ to $|0\rangle\langle 0|_{\mathcal{Z}} \otimes \boldsymbol{\rho}_{\mathcal{I}}$.

3. Apply $U(r)$ to $\mathcal{Z} \otimes \mathcal{I}$, measure $\mathcal{Z}$ in the computational basis, and return the outcome z.

A quantum oracle algorithm S^A with *black-box access* to $(A, \boldsymbol{\rho})$ does not have direct access to the adversary's *internal registers* $\mathcal{I}$, and can only operate on the state $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{I})$ by applying $U(r)$ or $U(r)^\dagger$ for any r. In more detail, black-box access to $(A, \boldsymbol{\rho})$ means the following:

- The registers $\mathcal{Z} \otimes \mathcal{I}$ are initialized to $|0\rangle\langle 0|_{\mathcal{Z}} \otimes \boldsymbol{\rho}_{\mathcal{I}}$.

- Once the $\mathcal{Z} \otimes \mathcal{I}$ registers are initialized, the algorithm is permitted to perform arbitrary operations on the $\mathcal{Z}$ register, but can only act on the $\mathcal{I}$ registers by applying $U(r)$ or $U(r)^\dagger$ for any r. We explicitly permit the $U(r)$ and $U(r)^\dagger$ gates to be controlled on any external registers (i.e., any registers other than the registers $\mathcal{Z} \otimes \mathcal{I}$ to which $U(r)$ is applied).

We note that this definition is consistent with the notions of interactive quantum machines and oracle access to an interactive quantum machine used in e.g. [Unr12] and other works on post-quantum zero-knowledge.

The Binary Input Case. Following [Mah18], in the special case where $r \in \{0,1\}$, it will be convenient to re-define the internal state to be $\boldsymbol{\rho} := U(0)(|0\rangle\langle 0|_{\mathcal{Z}} \otimes \boldsymbol{\rho}'_{\mathcal{I}})U(0)^\dagger$ (where $\boldsymbol{\rho}'_{\mathcal{I}} \in \mathrm{D}(\mathcal{I})$ denotes the "original" internal state), so that the behavior of A on $r = 0$ is to simply measure $\mathcal{Z}$ in the computational basis, and on $r = 1$ it applies the unitary $U := U(1)U(0)^\dagger$ to its state and then measures the $\mathcal{Z}$ register. Notice that in this case, the internal state is technically on $\mathcal{Z} \otimes \mathcal{I}$ instead of just $\mathcal{I}$. Thus, black-box access to a quantum algorithm with binary input is formalized as follows:

- The registers $\mathcal{Z} \otimes \mathcal{I}$ are initialized to $\boldsymbol{\rho} := U(0)(|0\rangle\langle 0|_{\mathcal{Z}} \otimes \boldsymbol{\rho}'_{\mathcal{I}})U(0)^\dagger$.
- Once the $\mathcal{Z} \otimes \mathcal{I}$ registers are initialized, the algorithm is permitted to perform arbitrary operations on the $\mathcal{Z}$ register, but can only act on the $\mathcal{I}$ registers by applying (possibly controlled) U or $U^\dagger$ gates.

In this special case, an algorithm with black-box access to A is denoted $S^{U,\boldsymbol{\rho}}$.

We remark that these definitions are tailored to the two-message challenge-response setting, whereas the protocols we consider in this paper have more rounds of interaction. However, our analysis will typically focus on a single back-and-forth round of interaction (e.g., the last two messages of the [Mah18] protocol), so $\boldsymbol{\rho}$ will be the intermediate state of the interactive algorithm right before the next challenge is sent.[13] Moreover, the unitaries $\{U(r)\}_r$ can be treated as independent of the (classical) protocol transcript before challenge r is sent, since we can assume this transcript is saved in $\boldsymbol{\rho}$.

3.3 Interactive Arguments

In what follows we define the notion of an interactive argument for **QMA** languages. We denote such arguments by (P, V), and denote the output bit of the verifier by $\mathsf{Out}(P, V)$.

Definition 3.1. *An interactive argument (P, V) for a language $\mathcal{L} = (\mathcal{L}_{\mathsf{yes}}, \mathcal{L}_{\mathsf{no}}) \in$ **QMA** with relation $\mathcal{R}(x)$ is a (classical) 2-party interactive protocol between a QPT prover P and a p.p.t. verifier V, with the following completeness and soundness guarantees:*

[13] In the multi-round setting, "re-defining" the intermediate state to be $\boldsymbol{\rho} = U(0)(|0\rangle\langle 0|_{\mathcal{Z}} \otimes \boldsymbol{\rho}'_{\mathcal{I}})U(0)^\dagger$ can be implemented by replacing any unitary W applied in the previous round with $U(0)W$; this follows the conventions used in [Mah18].

Completeness. *For all $\lambda \in \mathbb{N}$, there exists a polynomial $k = k(\lambda)$ such that for all $x \in \mathcal{L}_{yes}$, and all $|\phi\rangle \in \mathcal{R}(x)$, it holds that*

$$\Pr\left[\mathsf{Out}\left(P(|\phi\rangle^{\otimes k(\lambda)}, x), V(x)\right) = 1\right] \geq 1 - \mathsf{negl}(\lambda).$$

Computational Soundness. *For all $\lambda \in \mathbb{N}$, all $x \in \mathcal{L}_{\mathsf{no}}$, and all non-uniform QPT provers P^*, it holds that*

$$\Pr\left[\mathsf{Out}(P^*(x), V(x)) = 1\right] \leq \mathsf{negl}(\lambda).$$

Batch Arguments. We also consider a sub-class of interactive arguments where the prover simultaneously engages the verifier on n sub-instances $(x_1, \ldots, x_n)$, where each x_i is supposed to be a Yes-instance of a fixed language $\mathcal{L}_i$. We require the following notion of (computational) soundness.

Definition 3.2 (Soundness). *An interactive argument (P, V) for a batch language $\mathcal{L} = \mathcal{L}_1 \times \ldots \times \mathcal{L}_n \in$ **QMA** with relation $\mathcal{R}(x)$ is sound if for all $\lambda \in \mathbb{N}$, all polynomials $n = n(\lambda)$, all indices $i \in [n]$, all statements $(x_1, \ldots, x_n)$, where $x_i \in \mathcal{L}_{\mathsf{no}}$, and all non-uniform QPT provers P^*, it holds that*

$$\Pr\left[\mathsf{Out}(P^*(x_1, \ldots, x_n), V((x_1, \ldots, x_n))) = 1\right] \leq \mathsf{negl}(\lambda).$$

3.4 Computational Indistinguishability

Two classical distribution ensembles $\{(X^{(\lambda)}, Y^{(\lambda)})\}_\lambda$ are said to be post-quantum computationally indistinguishable if for every non-uniform QPT algorithm $A = \{(A^{(\lambda)}, \boldsymbol{\rho}^{(\lambda)})\}_\lambda$ (that outputs a bit b), we have that

$$\left|\mathbb{E}\left[A^{(\lambda)}(X^{(\lambda)}, \boldsymbol{\rho}^{(\lambda)})\right] - \mathbb{E}\left[A^{(\lambda)}(Y^{(\lambda)}, \boldsymbol{\rho}^{(\lambda)})\right]\right| = \mathsf{negl}(\lambda).$$

Two quantum state ensembles $\{\boldsymbol{\rho}_0^{(\lambda)}, \boldsymbol{\rho}_1^{(\lambda)}\}_\lambda$ are said to be *computationally indistinguishable* if for every non-uniform QPT algorithm $A = \{A^{(\lambda)}, \boldsymbol{\rho}^{(\lambda)}\}$ (that outputs a bit b), we have that

$$\left|\mathbb{E}\left[A^{(\lambda)}(\boldsymbol{\rho}^{(\lambda)}, \boldsymbol{\rho}_0^{(\lambda)})\right] - \mathbb{E}\left[A^{(\lambda)}(\boldsymbol{\rho}^{(\lambda)}, \boldsymbol{\rho}_1^{(\lambda)})\right]\right| = \mathsf{negl}(\lambda).$$

Equivalently, $\{\boldsymbol{\rho}_0^{(\lambda)}, \boldsymbol{\rho}_1^{(\lambda)}\}_\lambda$ are computationally indistinguishable if for every efficiently computable non-uniform binary observable $(R, \boldsymbol{\sigma})$, we have that

$$\left|\operatorname{Tr}(R(\boldsymbol{\rho}_0 \otimes \boldsymbol{\sigma})) - \operatorname{Tr}(R(\boldsymbol{\rho}_1 \otimes \boldsymbol{\sigma}))\right| = \mathsf{negl}(\lambda).$$

We will occasionally use the notation $\boldsymbol{\rho}_0 \approx_c \boldsymbol{\rho}_1$ to denote computational indistinguishability of $\{\boldsymbol{\rho}_0^{(\lambda)}, \boldsymbol{\rho}_1^{(\lambda)}\}_\lambda$.

More generally, we use $(T(\lambda), \varepsilon(\lambda))$-indistinguishability to denote computational indistiguishability as above where the distinguisher is allowed to run in time T and the advantage is required to be at most ε.

3.5 Mahadev Randomized TCFs

In this section, we define the cryptographic primitive used by Mahadev [Mah18] to obtain a (non-succinct) delegation scheme for **QMA** with classical verification. The primitive is closely related to Regev encryption [Reg05] and LWE-based "lossy" trapdoor functions [PW08, PVW08, GVW15], but makes use of special-purpose structure relevant for quantum functionality. Most of this special-purpose structure, in particular, the "adaptive hardcore bit", was introduced in the work of Brakerski, Christiano, Mahadev, Vazirani and Vidick [BCM$^+$18], but [Mah18] further requires "dual-mode key generation" in addition to the [BCM$^+$18] properties. Given the numerous special-purpose requirements, we refer to the primitive as "Mahadev randomized trapdoor claw-free functions (rTCFs)."[14]

Definition 3.3. *A Mahadev randomized trapdoor claw-free function family (Mahadev rTCF)* $\mathsf{ClawFree}$ *is described by a tuple of efficient classical algorithms* $(\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert}, \mathsf{Check}, \mathsf{Good})$ *with the following syntax:*

- $\mathsf{Gen}(1^\lambda, \mathsf{mode})$ *is a dual-mode PPT key generation algorithm that takes as input a security parameter* λ *in unary, and a bit* $\mathsf{mode} \in \{0,1\}$*, and it outputs a public key* pk *and a private key* sk*. The description of the public key implicitly defines a domain of the form* $\{0,1\} \times \mathcal{D}_{\mathsf{pk}}$ *for the randomized function* f_{pk}*. We view* $\mathcal{D}_{\mathsf{pk}}$ *as an explicit (efficiently verifiable and samplable) subset of* $\{0,1\}^{\ell(\lambda)}$*, so that applying bit operations to elements of* $\mathcal{D}_{\mathsf{pk}}$ *is well-defined.*

  In our context, $\mathsf{mode} = 0$ *samples keys for an* injective *function and* $\mathsf{mode} = 1$ *samples keys for a* two-to-one *function. For the sake of readability, we use a descriptive notation by which* $\mathsf{mode} \in \{\mathsf{injective}, \mathsf{2\text{-}to\text{-}1}\}$*, where* $\mathsf{mode} = \mathsf{injective}$ *corresponds to* $\mathsf{mode} = 0$ *and* $\mathsf{mode} = \mathsf{2\text{-}to\text{-}1}$ *corresponds to* $\mathsf{mode} = 1$*.*

- $\mathsf{Eval}(\mathsf{pk}, b, \mathbf{x})$ *is a (possibly probabilistic) algorithm that takes as input a public key* pk*, a bit* $b \in \{0,1\}$ *and an element* $\mathbf{x} \in \mathcal{D}_{\mathsf{pk}}$*, and outputs a string* $\mathbf{y}$ *with distribution* χ*.*

- $\mathsf{Invert}(\mathsf{mode}, \mathsf{sk}, \mathbf{y})$ *is a deterministic algorithm that takes as input* $\mathsf{mode} \in \{\mathsf{injective}, \mathsf{2\text{-}to\text{-}1}\}$*, a secret key* sk*, and an element* $\mathbf{y}$ *in the range. If* $\mathsf{mode} = \mathsf{injective}$ *then it outputs a pair* $(b, \mathbf{x}) \in \{0,1\} \times \mathcal{D}_{\mathsf{pk}}$ *or* $\perp$*. If* $\mathsf{mode} = \mathsf{2\text{-}to\text{-}1}$ *then it outputs two pairs* $(0, \mathbf{x}_0)$ *and* $(1, \mathbf{x}_1)$ *with* $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{D}_{\mathsf{pk}}$*, or* $\perp$*.*

- $\mathsf{Check}(\mathsf{pk}, b, \mathbf{x}, \mathbf{y})$ *is a deterministic algorithm takes as input a public key* pk*, a bit* $b \in \{0,1\}$*, an element* $\mathbf{x} \in \mathcal{D}_{\mathsf{pk}}$*, and an element* $\mathbf{y}$ *in the range, and it outputs a bit.*

- $\mathsf{Good}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{d})$ *is a deterministic poly-time algorithm that takes as input two domain elements* $\mathbf{x}_0, \mathbf{x}_1 \in \mathcal{D}_{\mathsf{pk}}$ *and a string* $\mathbf{d} \in \{0,1\}^{\ell+1}$*. It outputs a bit that characterizes membership in a set that we call*

$$\mathsf{Good}_{\mathbf{x}_0,\mathbf{x}_1} := \{\mathbf{d} : \mathsf{Good}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{d}) = 1\}.$$

[14] Actually, [Mah18] requires an extra (second) hardcore bit property (Property 2 of Definition 4.4 in [Mah18]) that we drop from our definition, as our proof does not require it.

Moreover, we stipulate that $\mathsf{Good}(\mathbf{x}_0, \mathbf{x}_1, \mathbf{d})$ *ignores the first bit of* $\mathbf{d}$*.*[15]

We require that the following properties are satisfied.

1. **Correctness:**

   (a) *For* all $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\mathsf{injective}, 1^\lambda)$*: For every* $b \in \{0,1\}$*, every* $\mathbf{x} \in \mathcal{D}_{\mathsf{pk}}$*, and every* $\mathbf{y} \in \mathsf{Supp}(\mathsf{Eval}(\mathsf{pk}, (b, \mathbf{x})))$*,*

   $$\mathsf{Invert}(\mathsf{injective}, \mathsf{sk}, \mathbf{y}) = (b, \mathbf{x}).$$[16]

   (b) *For* all $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\text{2-to-1}, 1^\lambda)$*: For every* $b \in \{0,1\}$*, every* $\mathbf{x} \in \mathcal{D}_{\mathsf{pk}}$*, and every* $\mathbf{y} \in \mathsf{Supp}(\mathsf{Eval}(\mathsf{pk}, (b, \mathbf{x})))$*,*

   $$\mathsf{Invert}(\text{2-to-1}, \mathsf{sk}, \mathbf{y}) = ((0, \mathbf{x}_0), (1, \mathbf{x}_1))$$

   such that $\mathbf{x}_b = \mathbf{x}$ *and* $\mathbf{y} \in \mathsf{Supp}(\mathsf{Eval}(\mathsf{pk}, (\beta, \mathbf{x}_\beta)))$ *for every* $\beta \in \{0,1\}$*.*

   (c) *For every* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Supp}(\mathsf{Gen}(\text{2-to-1}, 1^\lambda)) \cup \mathsf{Supp}(\mathsf{Gen}(\mathsf{injective}, 1^\lambda))$*, every* $b \in \{0,1\}$ *and every* $\mathbf{x} \in \mathcal{D}$*,*

   $$\Pr[\mathsf{Check}(\mathsf{pk}, (b, \mathbf{x}), \mathbf{y}) = 1] = 1$$

   if and only if $\mathbf{y} \in \mathsf{Supp}(\mathsf{Eval}(\mathsf{pk}, (b, \mathbf{x})))$*.*

   (d) *For every* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{Gen}(\text{2-to-1}, 1^\lambda)$ *and every pair of domain elements* $\mathbf{x}_0, \mathbf{x}_1$*, the density of* $\mathsf{Good}_{\mathbf{x}_0, \mathbf{x}_1}$ *is* $1 - \mathsf{negl}(\lambda)$*.*

2. **Key Indistinguishability:**

   $$\{\mathsf{pk} : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\text{2-to-1}, 1^\lambda)\} \approx_c \{\mathsf{pk} : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{injective}, 1^\lambda)\}$$

3. **Adaptive Hardcore Bit:** *For every* BQP *adversary* $\mathcal{A} = (\mathcal{A}_\prime, \mathcal{A}_\infty)$ *there exists a negligible function* μ *such that for every* $\lambda \in \mathbb{N}$*, the following difference of probabilities is equal to* $\mu(\lambda)$*:*

   $$\Big| \Pr[\mathcal{A}_1(\mathsf{pk}, \mathbf{y}) = (\mathbf{d}, (b, \mathbf{x})) : \mathsf{Check}(\mathsf{pk}, b, \mathbf{x}, \mathbf{y}) = 1 \quad \wedge \quad \mathbf{d} \cdot (1, \mathbf{x}_0 \oplus \mathbf{x}_1) = 0 \quad \wedge \quad \mathbf{d} \in \mathsf{Good}_{\mathbf{x}_0, \mathbf{x}_1}]$$
   $$- \Pr[\mathcal{A}_1(\mathsf{pk}, \mathbf{y}) = (\mathbf{d}, (b, \mathbf{x})) : \mathsf{Check}(\mathsf{pk}, b, \mathbf{x}, \mathbf{y}) = 1 \quad \wedge \quad \mathbf{d} \cdot (1, \mathbf{x}_0 \oplus \mathbf{x}_1) = 1 \quad \wedge \quad \mathbf{d} \in \mathsf{Good}_{\mathbf{x}_0, \mathbf{x}_1}] \Big|$$

   where the probabilities are over the experiment that generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\text{2-to-1}, 1^\lambda)$*,* $\mathbf{y} \leftarrow \mathcal{A}_0(\mathsf{pk})$*, and where* $((0, \mathbf{x}_0), (1, \mathbf{x}_1)) = \mathsf{Invert}(\text{2-to-1}, \mathsf{sk}, \mathbf{y})$*.*

Lemma 3.4 ([BCM+18, Mah18])**.** *Assuming LWE, there is a collection of Mahadev randomized TCFs.*

[15] We depart slightly from notation in prior work, which defines $\mathbf{d}$ to be an element of $\{0,1\}^\ell$ (corresponding to the last ℓ bits of our $\mathbf{d}$).

[16] Note that this implies that $\mathsf{Supp}(\mathsf{Eval}(\mathsf{pk}, (b_1, \mathbf{x}_1))) \cap \mathsf{Supp}(\mathsf{Eval}(\mathsf{pk}, (b_2, \mathbf{x}_2))) = \emptyset$ for every $(b_1, \mathbf{x}_1) \neq (b_2, \mathbf{x}_2)$. This can be enforced in the LWE-based instantiation by using *truncated* discrete Gaussian errors

Remark 3.5. *For some of our applications (and for simplicity of proofs), we will actually require an rTCF that is* perfectly correct, *which means that the correctness properties (a) and (b) hold with probability 1. That is, they hold for all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(\mathsf{injective}, 1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(\text{2-to-1}, 1^\lambda)$ *respectively. We briefly argue that this is possible. In the injective mode case, this is possible because the sampling procedure for injective keys given in [Mah18, Section 9.2] can determine whether the key it sampled is indeed injective and if not, output a fixed hard-coded injective key. In the 2-to-1 mode case, the sampling procedure given in [BCM+18, Section 4.1] is perfect except for when* $s = 0^n$*. Thus, we can again hard-code a fixed 2-to-1 key to output instead whenever* $s = 0^n$*.*

3.6 Collapsing Hash Functions

Collapsing Hash Functions. Let $H = \{H_\lambda\}_{\lambda \in \mathbb{N}}$ be a hash function family where each H_λ is a distribution over functions $h : \{0,1\}^{n(\lambda)} \to \{0,1\}^{\ell(\lambda)}$.

Define the collapsing experiment $\texttt{CollapseExpt}_{H,\lambda,b}(D)$ on quantum distinguisher D as follows. $\texttt{CollapseExpt}_{H,\lambda,b}(D)$:

1. The challenger samples $h \leftarrow H_\lambda$ and sends h to the distinguisher D.
2. The distinguisher replies with a classical binary string $y \in \{0,1\}^{\ell(\lambda)}$ and an $n(\lambda)$-qubit quantum state on the register $\mathcal{X}$. Note that the requirement that y be classical can be enforced by having the challenger immediately measure these registers upon receiving them.
3. The challenger computes h in superposition on the $n(\lambda)$-qubit quantum state, and measures the bit indicating whether the output of h equals y. If the output does not equal y, the challenger aborts and outputs $\perp$.
4. If $b = 0$, the challenger does nothing. If $b = 1$, the challenger measures the $n(\lambda)$-qubit state in the standard basis.
5. The challenger returns the contents of the $\mathcal{X}$ register to the distinguisher.
6. The distinguisher outputs a bit b'.

Definition 3.6 ([Unr16b]). $H = \{H_\lambda\}_\lambda$ *is* collapsing *if for every security parameter* $\lambda \in \mathbb{N}$ *and any polynomial-size quantum distinguisher* $\mathcal{D} = \{D_\lambda\}_\lambda$*, there exists a negligible function* μ *such that*

$$\left|\Pr\left[\texttt{CollapseExpt}_{H,\lambda,0}(D_\lambda) = 1\right] - \Pr\left[\texttt{CollapseExpt}_{H,\lambda,1}(D_\lambda) = 1\right]\right| \le \mu(\lambda).$$

Unruh [Unr16a] constructs collapsing hash functions from lossy functions, which can be based on LWE [PW08].

Lemma 3.7 ([PW08, Unr16a]). *Assuming LWE, a family of collapsing hash functions* $\{H_\lambda : \{0,1\}^* \to \{0,1\}^\lambda\}_\lambda$ *exists.*

3.7 Fully Homomorphic Encryption

We define fully homomorphic encryption (FHE), which is used in Section 9. A fully homomorphic encryption scheme $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ for (classical) polynomial-time computation is a tuple of four PPT algorithms.

- $\mathsf{Gen}(1^\lambda)$ takes as input the security parameter and outputs a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Enc}(\mathsf{pk}, m)$ takes as input a message m and outputs a ciphertext ct.
- $\mathsf{Eval}(f, \mathsf{ct})$ takes as input a ciphertext ct corresponding to an n-bit plaintext as well as a function $f : \{0,1\}^n \to \{0,1\}$. It outputs a ciphertext ct_f.
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ takes as input the secret key and a ciphertext. It outputs a message.

We require the following properties.

- **Evaluation/Decryption Correctness:** for any (polynomial-size circuit) function $f : \{0,1\}^n \to \{0,1\}$ and any message $m \in \{0,1\}^n$, we have that

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(f, \mathsf{Enc}(\mathsf{pk}, m))) = f(m)$$

  with probability $1 - \mathsf{negl}(\lambda)$ over the parameter sampling.

- **Compactness:** we require that $\mathsf{FHE.Eval}(f, \mathsf{Enc}(\mathsf{pk}, m))$ has a fixed size $\mathsf{poly}(\lambda)$ independent of $|f|, |m|$.
- **Semantic Security:** For any pair of messages (m_0, m_1), we have that $(\mathsf{pk}, \mathsf{FHE.Enc}(\mathsf{pk}, m_0)) \approx_c (\mathsf{pk}, \mathsf{FHE.Enc}(\mathsf{pk}, m_1))$.

Theorem 3.8 ([Gen09, BV11, BGV12, BV14]). *Under circular-secure variants of the Learning with Errors assumption, there exists a fully homomorphic encryption scheme for all polynomial-time computable functions. If the circular LWE variant is post-quantum, then so is the FHE scheme.*

Under the standard LWE assumption, there exists a FHE scheme for all polynomial-size circuits of depth $d(\lambda)$, where the scheme has compactness $\mathsf{poly}(\lambda, d)$.

3.8 Indistinguishability Obfuscation

An indistinguishability obfuscator (iO) is an algorithm $\mathsf{i}\mathcal{O}$ that takes as input a circuit C and satisfies the following properties.

- **Functional Equivalence:** for any (polynomial-size) circuit circuit $C : \{0,1\}^n \to \{0,1\}^m$ and any input $x \in \{0,1\}^n$, we have that

$$\mathsf{i}\mathcal{O}(C)(x) = C(x).$$

- **Security:** For any pair of functionally equivalent circuits (C_0, C_1), we have that

$$\mathsf{i}\mathcal{O}(C_0) \approx_c \mathsf{i}\mathcal{O}(C_1).$$

We mention that, while some recent candidates for iO (such as [JLS21]) can be broken using quantum algorithms, others, such as [BGMZ18, CVW18, BDGM20, WW21, GP21, DQV$^+$21], are plausibly post-quantum secure. Furthermore, it will be convenient for us to assume iO with perfect correctness to simplify our analysis (in particular the argument in Section 10.3). We point out that this property is already satisfied by most candidates and can also be attained via generic transformations [BV17].

3.9 Puncturable PRFs

Definition 3.9 (Puncturable PRF [BW13, BGI14, KPTZ13, SW14]). *A puncturable PRF family is a family of functions*

$$\mathcal{F} = \left\{F_{\lambda,s} : \{0,1\}^{\nu(\lambda)} \to \{0,1\}^{\mu(\lambda)}\right\}_{\lambda\in\mathbb{N}, s\in\{0,1\}^{\ell(\lambda)}}$$

with associated (deterministic) polynomial-time algorithms $(\mathcal{F}.\mathsf{Eval}, \mathcal{F}.\mathsf{Puncture}, \mathcal{F}.\mathsf{PuncEval})$ *satisfying*

- *For all $x \in \{0,1\}^{\nu(\lambda)}$ and all $s \in \{0,1\}^{\ell(\lambda)}$, $\mathcal{F}.\mathsf{Eval}(s,x) = F_{\lambda,s}(x)$.*
- *For all distinct $x, x' \in \{0,1\}^{\nu(\lambda)}$ and all $s \in \{0,1\}^{\ell(\lambda)}$,*

$$\mathcal{F}.\mathsf{PuncEval}(\mathcal{F}.\mathsf{Puncture}(s,x), x') = \mathcal{F}.\mathsf{Eval}(s,x')$$

For ease of notation, we write $F_s(x)$ and $\mathcal{F}.\mathsf{Eval}(s,x)$ interchangeably, and we write $s\{x\}$ to denote $\mathcal{F}.\mathsf{Puncture}(s,x)$.

$\mathcal{F}$ is said to be (s,δ)-secure if for every $\{x^{(\lambda)} \in \{0,1\}^{\nu(\lambda)}\}_{\lambda\in\mathbb{N}}$, the following two distribution ensembles (indexed by λ) are $\delta(\lambda)$-indistinguishable to circuits of size $s(\lambda)$:

$$(S\{x^{(\lambda)}\}, F_S(x^{(\lambda)})) \text{ where } S \leftarrow \{0,1\}^{\ell(\lambda)}$$

and

$$(S\{x^{(\lambda)}\}, U) \text{ where } S \leftarrow \{0,1\}^{\ell(\lambda)},\ U \leftarrow \{0,1\}^{\mu(\lambda)}.$$

Theorem 3.10 ([GGM84, KPTZ13, BW13, BGI14, SW14]). *If {polynomially secure, subexponentially secure} one-way functions exist, then for all functions $\mu : \mathbb{N} \to \mathbb{N}$ (with $1^{\mu(\nu)}$ polynomial-time computable from 1^ν), and all $\delta : \mathbb{N} \to [0,1]$ with $\delta(\nu) \geq 2^{-\mathsf{poly}(\nu)}$, there are polynomials $\ell(\lambda), \nu(\lambda)$ and a {polynomially secure, $(\frac{1}{\delta(\nu(\lambda))}, \delta(\nu(\lambda)))$-secure} puncturable PRF family*

$$\mathcal{F}_\mu = \left\{F_{\lambda,s} : \{0,1\}^{\nu(\lambda)} \to \{0,1\}^{\mu(\nu(\lambda))}\}_{\lambda\in\mathbb{N}, s\in\{0,1\}^{\ell(\lambda)}}\right\}.$$

4 Commit-and-Measure Protocols

4.1 Defining Commit-and-Measure Protocols

In this section, we formalize the notion of a *commit-and-measure protocol*, which was informally described in [Mah18]. A commit-and-measure protocol enables a classical verifier to obtain the results of measuring, in the standard or Hadamard basis, each qubit of an N-qubit quantum state $\boldsymbol{\sigma}$ held by the prover. More precisely, the verifier encodes its choice of basis with a classical circuit $C : [N] = \{0,1\}^{\log N} \to \{0,1\}$, where $C(i) = b$ specifies the basis for the measurement of the ith qubit. We adopt the convention that $b = 0$ corresponds to the standard basis and $b = 1$ corresponds to the Hadamard basis. Note that in Mahadev's original protocol, C is given as an explicit string $(C(0), C(1), \ldots, C(N-1))$, but our eventual *succinct* protocols will require circuits C with size much smaller than N.

Definition 4.1 (Commit-and-Measure Protocol Syntax). *An N-qubit commit-and-measure protocol between a quantum polynomial-time prover $P = (\mathsf{Commit}, \mathsf{Open})$ and a classical probabilistic polynomial-time verifier $V = (\mathsf{Gen}, \mathsf{Test}, \mathsf{Out})$ has the following syntax.*

1. *The verifier samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)$, where $C : [N] = \{0,1\}^{\log N} \to \{0,1\}$ represents a* basis vector $h \in \{0,1\}^N$*, obtaining public parameters* pk *and secret parameters* sk*. It sends the public parameters* pk *to the prover.*

2. *The prover computes $(y, \boldsymbol{\rho}) \leftarrow \mathsf{Commit}(\mathsf{pk}, \boldsymbol{\sigma})$, obtaining a classical "commitment" string y and a private quantum state $\boldsymbol{\rho}$. It sends y to the verifier.*

3. *The verifier samples a random challenge bit $c \leftarrow \{0,1\}$ and sends c to the prover; $c = 0$ corresponds to a "test round" and $c = 1$ corresponds to a "measurement round".*

4. *The prover computes $z \leftarrow \mathsf{Open}(\boldsymbol{\rho}, c)$, obtaining a classical string z that it sends to the verifier.*

5. *If $c = 0$, the verifier computes $\{\mathsf{acc}, \mathsf{rej}\} \leftarrow \mathsf{Test}(\mathsf{pk}, (y, z))$.*

   If $c = 1$, the verifier computes $m \leftarrow \mathsf{Out}(\mathsf{sk}, (y, z))$ to obtain a classical string $m \in \{0,1\}^N$ of measurement outcomes.

The protocol is required to satisfy the following completeness (Definition 4.2) and soundness (Definition 4.5) properties. For the definitions below, we write $M(h, \boldsymbol{\sigma})$ to denote the distribution of outcomes from measuring $\boldsymbol{\sigma}$ in the basis h.

Definition 4.2 (Completeness). *A commit-and-measure protocol is required to satisfy two completeness properties.*

1. *(Test Round Completeness) For all $C : [N] \to \{0,1\}$ and N-qubit states $\boldsymbol{\sigma}$:*

$$\Pr\left[\mathsf{acc} \leftarrow \mathsf{Test}(\mathsf{pk}, (y, z)) : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C) \\ (y, \boldsymbol{\rho}) \leftarrow \mathsf{Commit}(\mathsf{pk}, \boldsymbol{\sigma}) \\ z \leftarrow \mathsf{Open}(\boldsymbol{\rho}, 0) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

2. *(Measurement Round Completeness) For all $C : [N] \to \{0,1\}$ and N-qubit states $\boldsymbol{\sigma}$:*

$$\left\{ m \leftarrow \mathsf{Out}(\mathsf{sk}, (y,z)) : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C) \\ (y, \boldsymbol{\rho}) \leftarrow \mathsf{Commit}(\mathsf{pk}, \boldsymbol{\sigma}) \\ z \leftarrow \mathsf{Open}(\boldsymbol{\rho}, 1) \end{array} \right\} \approx_c M(h, \boldsymbol{\sigma}),$$

where $h \in \{0,1\}^N$ is such that $h_i = C(i)$ for all $i \in [N] = \{0,1\}^{\log N}$.

Remark 4.3. *The [Mah18] protocol satisfies* statistical *measurement round completeness, but our verifier-succinct commit-and-measure protocol will not.*

Remark 4.4. *One of our applications will require a measurement protocol with* perfect *completeness, which stipulates that the above completeness guarantees hold over* all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(1^\lambda, C)$ *(and where the measurement round completeness is statistical rather than computational). This can be achieved by using an rTCF with perfect correctness, which we discuss in Section 3.5, and, in the succinct case, an indistinguishability obfuscation scheme with perfect correctness (Section 3.8).*

To state our soundness definition (Definition 4.5), we first specify the registers that any non-uniform cheating prover acts on:

- $\mathcal{P}$ contains the public parameters pk,
- $\mathcal{Y}$ contains the classical commitment string y,
- $\mathcal{Z}$ contains the classical opening string z,
- $\mathcal{I}$ contains the prover's initial state and its internal work registers.

In a protocol execution, $\mathcal{P}$ is initialized with $|\mathsf{pk}\rangle\langle\mathsf{pk}|$. A non-uniform cheating prover

$$P^* = (\boldsymbol{\rho}_0, U_{\mathsf{Commit}^*}, U_{\mathsf{Open}^*,0}, U_{\mathsf{Open}^*,1})$$

is parameterized by:

- An arbitrary quantum state $\boldsymbol{\rho}_0 \in \mathrm{D}(\mathcal{Y} \otimes \mathcal{Z} \otimes \mathcal{I})$. In a protocol execution with $\widetilde{P}$, $\mathcal{Y} \otimes \mathcal{Z} \otimes \mathcal{I}$ is initialized with $\boldsymbol{\rho}_0$.
- An adversarial commitment unitary U_{Commit^*} on $\mathcal{P} \otimes \mathcal{Y} \otimes \mathcal{Z} \otimes \mathcal{I}$ of the form

  $$\sum_{\mathsf{pk}} |\mathsf{pk}\rangle\langle\mathsf{pk}|_{\mathcal{P}} \otimes (U_{\mathsf{Commit}^*,\mathsf{pk}})_{\mathcal{Y},\mathcal{Z},\mathcal{I}}.$$

  That is, U_{Commit^*} is classically controlled on $\mathcal{P}$. In particular, the adversarial prover's commitment on verifier message pk is obtained by measuring register $\mathcal{Y}$ of

  $$U_{\mathsf{Commit}^*}(|\mathsf{pk}\rangle\langle\mathsf{pk}|_{\mathcal{P}} \otimes (\boldsymbol{\rho}_0)_{\mathcal{Y},\mathcal{Z},\mathcal{I}})$$

  in the computational basis to obtain y.

- An adversarial opening unitary $U_{\mathsf{Open}^*,0}$ on $\mathcal{P} \otimes \mathcal{Y} \otimes \mathcal{Z} \otimes \mathcal{I}$ corresponding to the prover's behavior in the test round $(b = 0)$ of the form:

  $$\sum_{\mathsf{pk},y} |\mathsf{pk}, y\rangle\langle \mathsf{pk}, y|_{\mathcal{P},\mathcal{Y}} \otimes (U_{\mathsf{Open}^*,0,\mathsf{pk},y})_{\mathcal{Z},\mathcal{I}}.$$

  That is, $U_{\mathsf{Open}^*,0}$ is classically controlled on $\mathcal{P}$ and $\mathcal{Y}$. In particular, given a commitment string y and residual prover state $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{Z} \otimes \mathcal{I})$, the prover's response on challenge $c = 0$ is obtained by measuring register $\mathcal{Z}$ of

  $$U_{\mathsf{Open}^*,0}(|\mathsf{pk}, y\rangle\langle \mathsf{pk}, y|_{\mathcal{P},\mathcal{Y}} \otimes \boldsymbol{\rho}_{\mathcal{Z},\mathcal{I}})$$

  in the computational basis to obtain z.

- An adversarial opening unitary $U_{\mathsf{Open}^*,1}$ on $\mathcal{P} \otimes \mathcal{Y} \otimes \mathcal{Z} \otimes \mathcal{I}$ corresponding to the prover's behavior in the measurement round $(b = 1)$ of the form:

  $$\sum_{\mathsf{pk},y} |\mathsf{pk}, y\rangle\langle \mathsf{pk}, y|_{\mathcal{P},\mathcal{Y}} \otimes (U_{\mathsf{Open}^*,1,\mathsf{pk},y})_{\mathcal{Z},\mathcal{I}}.$$

  In particular, given a commitment string y and residual prover state $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{Z} \otimes \mathcal{I})$, the prover's response on challenge $c = 1$ is obtained by measuring register $\mathcal{Z}$ of

  $$U_{\mathsf{Open}^*,1}(|\mathsf{pk}, y\rangle\langle \mathsf{pk}, y|_{\mathcal{P},\mathcal{Y}} \otimes \boldsymbol{\rho}_{\mathcal{Z},\mathcal{I}})$$

  in the Hadamard basis to obtain z.

Following [Mah18], we can assume without loss of generality that $U_{\mathsf{Open}^*,0}$ is the identity (refer to Section 3.2 for additional details). We will therefore write U to describe the prover's "attack unitary" for the measurement round $(c = 1)$.

For defining soundness, we informally require that a prover P^* that passes the test round with probability $1 - \mathsf{negl}(\lambda)$ (this could alternatively be enforced by applying a measurement in the security game) implicitly defines[17] an N-qubit state $\boldsymbol{\tau}$ whose measurement outcome distribution matches the output distribution of $\mathsf{Out}(\cdot)$ (up to computational indistinguishability).

Definition 4.5 (Soundness). *There exists an efficient classical algorithm* $\mathsf{SimGen}(1^\lambda)$ *and an efficient quantum algorithm* $\mathsf{Ext}^{U,\boldsymbol{\rho}}(\mathsf{pk}, \mathsf{sk}, y)$ *with black-box access to an attacker parameterized by a state* $\boldsymbol{\rho}$ *and a unitary* U *(see Section 3.2 for more details on how we formalize quantum black-box access), that takes as input classical strings* $(\mathsf{pk}, \mathsf{sk}, y)$*, and satisfies the following properties:*

- *Consider any non-uniform QPT cheating prover* $P^* = (\boldsymbol{\rho}_0, U_{\mathsf{Commit}^*}, U)$ *that passes the test round with probability* $1 - \mathsf{negl}(\lambda)$ *for all* $h \in \{0,1\}^N$*.*

  Then, for all $h \in \{0,1\}^N$ *with circuit representation* C*, the following two distributions are computationally indistinguishable:*

  Real:

[17] In fact, we require that $\boldsymbol{\tau}$ can be extracted efficiently from P^*.

1. *Sample parameters* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)$*.*
2. *Run the attacker* P^* *on* pk *to obtain a classical commitment string* y *(i.e., apply* U_{Commit^*} *and then measure the register containing* y*). Denote the post-measurement state as* $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{Z} \otimes \mathcal{I})$*, where* $\mathcal{Z}$ *corresponds to the registers that will eventually be measured to obtain the prover's final message, and* $\mathcal{I}$ *contains all of the other internal registers of the prover.*[18]
3. *Apply the prover's attack unitary* U*. This yields the state* $\boldsymbol{\rho}' := U\boldsymbol{\rho}_{\mathcal{Z},\mathcal{I}}U^\dagger$*. Measure the* $\mathcal{Z}$ *register of* $\boldsymbol{\rho}'$ *in the Hadamard basis to obtain the prover's opening string* z*.*
4. *Compute* $m \leftarrow \mathsf{Out}(\mathsf{sk}, (y, z))$ *and output* m*.*

Sim:

1. *Sample parameters* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{SimGen}(1^\lambda)$*.*
2. *Run the attacker* P^* *on* pk *to obtain a classical commitment string* y *(i.e., apply* U_{Commit^*} *and then measure the register containing* y*). Denote the post-measurement state as* $\boldsymbol{\rho} \in \mathrm{D}(\mathcal{Z} \otimes \mathcal{I})$*.*
3. *Run* $\mathsf{Ext}^{U,\boldsymbol{\rho}}(\mathsf{pk}, \mathsf{sk}, y) \to \boldsymbol{\tau}$ *to obtain an* N*-qubit state* $\boldsymbol{\tau}$*.*
4. *Measure each qubit of* $\boldsymbol{\tau}$ *according to the bases specified by* $h \in \{0,1\}^N$ *(i.e., qubit* i *is measured in the Hadamard basis if* $h_i = 1$ *and the standard basis if* $h_i = 0$*) and output the result.*

5 A Measurement Protocol Template

In this section, we describe a generic construction of a N-qubit commit-and-measure protocol (Section 4) using two building blocks: (1) a family of Mahadev rTCFs (Definition 3.3), and (2) a "batch key generation" scheme (fully defined in Section 7) whose syntax we describe below. We consider two different instantiations of this template:

- Using a "trivial" batch key generation scheme in which the N rTCF keys are sampled i.i.d., we recover Mahadev's original protocol [Mah18].
- Using a succinct key generation scheme (constructed in Sections 7 and 7.2, we obtain a measurement protocol in which the verifier's messages are succinct. We refer to this as a **verifier-succinct** measurement protocol.

Batch Key Generation. For our construction, we make use of what we call a "batch key generation scheme" for the Mahadev rTCF. Let $\mathsf{TCF.Gen}(1^\lambda, \mathsf{mode})$ denote the "standard" key generation algorithm for a Mahadev rTCF. Informally, a batch key generation scheme for $\mathsf{TCF.Gen}(1^\lambda, \mathsf{mode})$ is a mechanism that produces a joint representation of N TCF pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$, from which any

[18] We will also assume, without loss of generality, that the prover always copies pk and y into its internal state registers $\mathcal{I}$.

individual $\mathsf{pk}_i, \mathsf{sk}_i$ can be computed, such that the pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ are sufficiently "independent" of each other.

A full definition of a batch key generation scheme is given in Definition 7.2, but we formally state here the relevant syntax and security properties. Syntactically, a batch key generation scheme includes three algorithms $(\mathsf{Gen}, \mathsf{ExtPk}, \mathsf{ExtSk})$, where:

- $\mathsf{Gen}(1^\lambda, C)$ takes as input a security parameter λ and a circuit $C : [N] \to \{0,1\}$ representing (through its truth table) an N-bit string. It outputs a master public key PK and master secret key SK.

- $\mathsf{ExtPk}(\mathsf{PK}, i)$ is a deterministic algorithm that takes as input PK and an index $i \in N$, and outputs a public key pk_i.

- $\mathsf{ExtSk}(\mathsf{SK}, i)$ is a deterministic algorithm that takes as input SK and an index $i \in N$, and outputs a secret key sk_i.

When instantiated for a Mahadev rTCF family, we require the following properties to hold for such a procedure:

- **Correctness**: for $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda, N, C)$ and $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathsf{ExtPk}(\mathsf{PK}, i), \mathsf{ExtSk}(\mathsf{SK}, i))$, we have that $(\mathsf{pk}_i, \mathsf{sk}_i)$ is in the range of $\mathsf{TCF}.\mathsf{Gen}(1^\lambda, C(i))$ (i.e. they are a valid key pair in mode $C(i)$).

- **Key Indistinguishability**: if C_1 and C_2 represent functions that agree on a set T of inputs, then PK output by $\mathsf{Gen}(1^\lambda, C_1)$ is computationally indistinguishable from PK output by $\mathsf{Gen}(1^\lambda, C_2)$, *even* in the presence of all $\{\mathsf{sk}_i, i \in T\}$.

- **Collapsing at a single index**: For any index j, the function f_{pk_j} is collapsing even given all secret keys sk_i for $i \neq j$.

- **Adaptive hardcore bit at a single index**: For any index j, the function f_{pk_j} satisfies the rTCF adaptive hardcore bit property even given all secret keys sk_i for $i \neq j$.

Our protocol is a variant of the Mahadev protocol [Mah18] in which the verifier's first message $(\mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ is replaced by the output PK of a batch key generation procedure.

5.1 Measurement Protocol Description

Let $\mathsf{ClawFree} = (\mathsf{TCF}.\mathsf{Gen}, \mathsf{Eval}, \mathsf{Invert}, \mathsf{Check}, \mathsf{Good})$ denote a family of [Mah18] randomized TCFs (Definition 3.3). For simplicity of the analysis, we assume that on a fixed security parameter 1^λ, Gen outputs keys defining functions from $\{0,1\} \times D$ to R, where D is a fixed-size subset of $\{0,1\}^{\ell(\lambda)}$ independent of pk.[19] We denote the size of D by $L(\lambda)$. Let $(\mathsf{Gen}, \mathsf{ExtPk}, \mathsf{ExtSk})$ denote a batch key generation algorithm for $\mathsf{TCF}.\mathsf{Gen}$ satisfying the requirements described above.

[19] The LWE-based instantiation from [BCM+18] satisfies this property. However, with more complicated expressions, our proof should go through without this simplifying assumption.

For simplicity, we assume the honest prover P has an N-qubit pure state of the form

$$|\psi\rangle_{\mathcal{W}} = \sum_{w_1,\ldots,w_N} \alpha_{w_1,\ldots,w_N} |w_1 \ldots w_N\rangle$$

on its internal N-qubit register $\mathcal{W}$; the protocol completeness immediately extends to mixed states by linearity. The verifier has a basis string $h \in \{0,1\}^N$, represented by a circuit $C : [N] \to \{0,1\}$, and wants to obtain a measurement of $|\psi\rangle$ in this basis (where $h_i = 0$ denotes a standard basis measurement and $h_i = 1$ denotes a Hadamard basis measurement). The protocol MP (for measurement protocol), with prover algorithm $P = (\mathsf{MP.Commit}, \mathsf{MP.Open})$ and verifier algorithm $V = (\mathsf{MP.Gen}, \mathsf{MP.Test}, \mathsf{MP.Out})$, is executed as follows.

- MP.Gen: On input 1^λ, N and C, the verifier samples a representation of N TCF key pairs by computing $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda, N, C)$, where C is circuit computing $C(i) = h_i$. The verifier sends PK to the prover.
- MP.Commit: Given PK and its input state $|\psi\rangle_{\mathcal{W}}$, the prover computes public keys $\mathsf{pk}_i \leftarrow \mathsf{ExtPk}(\mathsf{PK}, i)$ for $1 \le i \le N$. From now on, P operates directly on (pk_i) and ignores PK.

  The prover coherently (with respect to $|\psi\rangle$) computes a "range superposition"

  $$\frac{1}{\sqrt{L^N}} \sum_{\substack{x_1,\ldots,x_N \in D \\ y_1,\ldots,y_N \in R \\ w \in \{0,1\}^N}} \left(\alpha_w \prod_i \sqrt{p_{\mathsf{pk}_i}(w_i, x_i, y_i)}\, |w\rangle_{\mathcal{W}} |x_1\rangle_{\mathcal{X}_1} \ldots |x_N\rangle_{\mathcal{X}_N} |y_1\rangle_{\mathcal{Y}_1} \ldots |y_N\rangle_{\mathcal{Y}_N} \right)$$

  where each $\mathcal{X}_i$ is an $\ell(\lambda)$-qubit register (where $D \subset \{0,1\}^\ell$), and each $\mathcal{Y}_i$ has basis $\{|y\rangle\}_{y \in R}$. Here, $p_{\mathsf{pk}}(b,x,y)$ denotes the probability density of y in the distribution $f_{\mathsf{pk}}(b,x)$, where $p_{\mathsf{pk}}(b,x,y) := 0$ for $x \in \{0,1\}^\ell \setminus D$. Following [BCM+18, Section 4.3] the honest prover algorithm can efficiently prepare this state up to exponentially small trace distance.

  After preparing this state, the prover measures $\mathcal{Y}_1, \ldots, \mathcal{Y}_N$ in the standard (R-)basis and sends the outcome $(y_1, \ldots, y_N)$ to the verifier.
- The verifier sends a uniformly random challenge bit c. After receiving the prover response, the verifier computes each public key $\mathsf{pk}_i \leftarrow \mathsf{ExtPk}(\mathsf{PK}, i)$ and secret key $\mathsf{sk}_i \leftarrow \mathsf{ExtSk}(\mathsf{SK}, i)$ in order to evaluate either MP.Test or MP.Out.
- MP.Open: On challenge bit c, the prover operates as follows.
  - If $c = 0$, the prover measures $\mathcal{W} \otimes \mathcal{X}_1 \otimes \ldots \otimes \mathcal{X}_N$ in the standard basis and sends the outcome $(b_1, \ldots, b_N, x_1, \ldots, x_N)$ to the verifier.
  - If $c = 1$, the prover instead measures $\mathcal{W} \otimes \mathcal{X}_1 \otimes \ldots \otimes \mathcal{X}_N$ in the Hadamard basis, returning strings $d_1, \ldots d_N$.
- MP.Test: Given $(b_1, \ldots, b_N, x_1, \ldots, x_N)$, the verifier computes (for every i) $\mathsf{Check}(\mathsf{pk}_i, b_i, x_i, y_i)$ and rejects if any of these checks do not pass.
- MP.Out: Given $d_1, \ldots d_N$, the verifier outputs N bits as follows. For each $i \in [N]$:

- If $h_i = 0$, the verifier ignores d_i, computes $(b_i, x_i) = \mathsf{Invert}(\mathsf{injective}, \mathsf{sk}_i, y_i)$, and outputs b_i.
- If $h_i = 1$, the verifier computes the two inverses $\{(0, x_{0,i}), (1, x_{1,i})\} \leftarrow \mathsf{Invert}(\text{2-to-1}, \mathsf{sk}_i, y_i)$. For each i, the verifier checks whether $d_i \in \mathsf{Good}_{x_{0,i},x_{1,i}}$ (corresponding to a valid equation in the ith slot), and if so, the verifier outputs $d_i \cdot (1, x_{0,i} \oplus x_{1,i})$. If $d_i \notin \mathsf{Good}_{x_{0,i},x_{1,i}}$, the verifier samples a uniformly random bit and outputs it.

Completeness of this protocol follows immediately from [Mah18] and the correctness property of Gen. Specifically, the correctness property of Gen implies that each $(\mathsf{pk}_i, \mathsf{sk}_i)$ in our protocol is in the range of $\mathsf{TCF.Gen}(1^\lambda, C(i))$, in which case (as shown in [Mah18]) the verifier's output distribution is statistically close to $h = (C(0), \ldots, C(N))$-measurement outcome on $|\psi\rangle$.

6 Soundness of Mahadev's Protocol

In this section, we prove that the measurement protocol from Section 5 a computationally sound (Definition 4.5) commit-and-measure protocol. As a consequence, we obtain a new, self-contained proof of soundness of the [Mah18] protocol. Later (Section 7.1), we will instead instantiate our protocol with a *succinct* key generation algorithm to obtain a verifier-succinct measurement protocol.

Our soundness proof is based in part on both [Mah18] itself as well as a proof strategy suggested in [Vid20].

Notation. Throughout this section, we will fix the verifier's choice of basis $h \in \{0,1\}^N$. We write $R := \{i \in [N] : h_i = 1\}$ and $S := \{i \in [N] : h_i = 0\}$, where $R \subset [N]$ denotes the set of indices that the verifier wants to measure in the Hadamard basis, and $S \subset [N]$ denotes the set of indices the verifier wants to measure in the standard basis.

Finally, we will decompose the state space of the prover as $\mathcal{Z} \otimes \mathcal{I}$, where:

- $\mathcal{Z} = \mathcal{Z}_1 \otimes \cdots \otimes \mathcal{Z}_N$. $1 \leq i \leq N$, $\mathcal{Z}_i$ is an $(\ell(\lambda) + 1)$-qubit register that contains the classical opening string z_i. We will sometimes write $\mathcal{Z}$ as shorthand for $\mathcal{Z}_1 \otimes \ldots \otimes \mathcal{Z}_N$.
- Each $\mathcal{Z}_i$ can be written as $\mathcal{Z}_i = \mathcal{B}_i \otimes \mathcal{X}_i$, where $\mathcal{B}_i$ is a one-bit register and $\mathcal{X}_i$ is an $\ell(\lambda)$-bit register.
- $\mathcal{I}$ denotes any additional registers the prover uses.

6.1 The Verifier's Output Distribution

Our goal is to characterize the N-bit distribution $D_{P^*,\mathrm{Out}}$ corresponding to the verifier's output in the measurement protocol when interacting with a malicious prover P^* using $h \in \{0,1\}^N$ as its choice of bases. In particular, we want to prove that if P^* succeeds in passing the test round with probability $1 - \mathsf{negl}(\lambda)$, then $D_{P^*,\mathrm{Out}}$ is computationally indistinguishable from $D_{P^*,\mathrm{Ext}}$, a distribution obtained from (1) running an efficient extractor Ext^{P^*} to obtain an N-qubit quantum state $\boldsymbol{\tau}$, and (2) measuring $\boldsymbol{\tau}$ in the verifier's specified bases.

The distribution $D_{P^*,\mathrm{Out}}$ produces a sample according to the following steps:

1. Sample keys $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{MP.Gen}(1^\lambda, N, C)$ (where C specifies the choice of bases R, S).

2. Run the malicious prover on PK to obtain a classical commitment string y. Let $|\psi\rangle \in \mathcal{Z} \otimes \mathcal{I}$ denote the prover's residual state.

3. For each $i \in S$, compute $(b_i, x_i) \leftarrow \mathsf{Invert}(\mathsf{injective}, \mathsf{sk}_i, y_i)$. Let $v \in \{0,1\}^S$ be the vector whose ith entry is b_i.

4. Next, apply the prover's attack unitary U on $\mathcal{Z} \otimes \mathcal{I}$, and then measure $\mathcal{Z}$ in the Hadamard basis to obtain a response $z = (d_1, \ldots, d_N)$.

5. For each $i \in R$, compute $(0, x_{0,i}), (1, x_{1,i}) \leftarrow \mathsf{Invert}(\text{2-to-1}, \mathsf{sk}_i, y_i)$. If $d_i \in \mathsf{Good}_{x_{0,i}, x_{1,i}}$, set $u_i = d_i \cdot (1, x_{0,i} \oplus x_{1,i})$. Otherwise, set u_i to be a uniformly random bit. This results in a string $u \in \{0,1\}^R$

6. Output $(u, v) \in \{0,1\}^R \times \{0,1\}^S$.

Our first step is to show that $D_{P^*,\mathsf{Out}}$ is computationally indistinguishable from a distribution $D_{P^*,\text{2-to-1}}$ that does *not* require running the Invert algorithm for any key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ in injective mode. Instead, this second distribution can be produced by directly measuring the register $\mathcal{B}_i$ (i.e., the first bit of $\mathcal{Z}_i$) of the prover's state $|\psi\rangle$). Since $\{\mathsf{sk}_i\}_{i \in S}$ will no longer be required at this point, we are also able to switch all key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ to be in two-to-one mode by invoking key indistinguishability.

Formally, $D_{P^*,\text{2-to-1}}$ produces outcomes as follows (differences from $D_{P^*,\mathrm{Out}}$ highlighted in red):

1. Sample keys $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{MP.Gen}(1^\lambda, N, \mathbf{1})$ (where $\mathbf{1}$ denotes the constant 1 function, corresponding to two-to-one mode)

2. Run the malicious prover on PK to obtain a classical commitment string y. Let $|\psi\rangle \in \mathcal{Z} \otimes \mathcal{I}$ denote the prover's residual state.

3. For each $i \in S$, measure $\mathcal{B}_i$ to obtain a bit v_i; the result of this step is a string $v \in \{0,1\}^S$.

4. Next, apply the prover's attack unitary U on $\mathcal{Z} \otimes \mathcal{I}$, and then measure $\mathcal{Z}$ in the Hadamard basis to obtain a response $z = (d_1, \ldots, d_N)$.

5. For each $i \in R$, compute $(0, x_{0,i}), (1, x_{1,i}) \leftarrow \mathsf{Invert}(\text{2-to-1}, \mathsf{sk}_i, y_i)$. If $d_i \in \mathsf{Good}_{x_{0,i}, x_{1,i}}$, set $u_i = d_i \cdot (1, x_{0,i} \oplus x_{1,i})$. Otherwise, set u_i to be a uniformly random bit. This results in a string $u \in \{0,1\}^R$

6. Output $(u, v) \in \{0,1\}^R \times \{0,1\}^S$.

Lemma 6.1. *$D_{P^*,\text{2-to-1}}$ is computationally indistinguishable from $D_{P^*,\mathrm{Out}}$.*

We prove Lemma 6.1 by first switching the keys sampled in $D_{P^*,\text{2-to-1}}$ to match the verifier's basis choice h. That is, we define the distribution $D_{P^*,h}$ to be the same distribution as $D_{P^*,\text{2-to-1}}$, except that the keys $(\mathsf{pk}_i, \mathsf{sk}_i)$ are sampled in mode h_i, i.e., Step 1 is replaced with:

1. Sample keys $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{MP.Gen}(1^\lambda, N, C)$ (where $C(i) = h_i$ for all i).

This is well-defined because the ith bit of the output is still obtained by measuring $\mathcal{B}_i$, which can be done regardless of how $(\mathsf{pk}_i, \mathsf{sk}_i)$ is sampled.

Claim 6.2. *For every basis choice h, $D_{P^*,h}$ is computationally indistinguishable from $D_{P^*,\text{2-to-1}}$.*

Proof. This follows by invoking the following key indistinguishability property of Gen:

$$\left\{(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda, 1) : (\mathsf{PK}, \{\mathsf{sk}_i\}_{i \notin S})\right\} \approx_c \left\{(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Gen}(1^\lambda, C) : (\mathsf{PK}, \{\mathsf{sk}_i\}_{i \notin S})\right\}.$$

Since the distributions are sampled without use of sk_i for all $i \in S$, Claim 6.2 follows from this indistinguishability. □

To conclude that $D_{P^*,\text{2-to-1}} \approx_c D_{P^*,\mathrm{Out}}$, we note:

Claim 6.3. *If $D_{P^*,\mathsf{Out}}$ is instantiated with basis choice h, then $D_{P^*,\mathsf{Out}}$ is statistically indistinguishable from $D_{P^*,h}$.*

Proof. Claim 6.3 follows from the injectivity of f_{pk_i} for each $i \in S$; by the correctness of Gen, we have that each pk_i (for $i \in S$) is in the support of $\mathsf{TCF.Gen}(1^\lambda, \mathsf{injective})$. Therefore, since $|\psi\rangle$ is guaranteed to pass the test round with probability $1 - \mathsf{negl}(\lambda)$, we have that with probability $1 - \mathsf{negl}(\lambda)$, measuring $\mathcal{B}_i$ gives the same result as computing the first bit of $\mathsf{Invert}(\mathsf{sk}_j, y_j)$ (which is the verifier's output). □

6.2 The Protocol Observables

Defining the Protocol Observables. In $D_{P^*,\text{2-to-1}}$, the entire N-bit output (u, v) is the result of performing measurements on $|\psi\rangle$, the prover's residual state after it sends its commitment y.

We now define a collection of binary observables $\{X_i, Z_i\}_{i \in [N]}$, parameterized by $(\mathsf{PK}, \mathsf{SK}, y)$ and the malicious prover's attack unitary U, such that the following process is equivalent to sampling from $D_{P^*,\text{2-to-1}}$:

1. Sample keys $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{MP.Gen}(1^\lambda, N, \mathbf{1})$.
2. Run the malicious prover on PK to obtain a classical commitment string y. Let $|\psi\rangle$ denote the prover's residual state.
3. For each $i \in S$, measure $|\psi\rangle$ with the observable Z_i to obtain a bit v_i.
4. Next, for each $i \in R$, measure the observable X_i to obtain a bit u_i.
5. Output $(u, v) \in \{0,1\}^R \times \{0,1\}^S$.

The definition of the Z_i observable is straightforward: since each v_i is obtained by measuring $\mathcal{B}_i$ in the standard basis, Z_i is simply the Pauli-Z observable $Z_i := (\sigma_Z)_{\mathcal{B}_i}$.

Defining the X_i observable requires more care. In $D_{P^*,\text{2-to-1}}$, the string $u \in \{0,1\}^R$ is obtained by applying the following steps (after $v \in \{0,1\}^S$ is measured)

1. Apply the prover's attack unitary U on $\mathcal{Z} \otimes \mathcal{I}$.

2. For each $i \in R$:
    (a) Apply $H^{\otimes \ell+1}$ to the register $\mathcal{Z}_i$ containing the prover's response in the ith slot.
    (b) Measure $\mathcal{Z}_i$ to obtain d_i. If $d_i \in \mathsf{Good}(x_{0,i}, x_{1,i})$, set $u_i = d_i \cdot (1, x_{0,i} \oplus x_{1,i})$. If $d_i \notin \mathsf{Good}(x_{0,i}, x_{1,i})$, set u_i to be a uniformly random bit.

In order to output a uniformly random bit, we will prepare fresh one-qubit ancilla registers $\mathcal{U}_1, \ldots, \mathcal{U}_N$, so that in the event that the prover returns an invalid d_i in slot i, the verifier can generate a random bit by measuring $\mathcal{U}_i$ (initialized to $|0\rangle$) in the Hadamard basis. Note that the $\mathcal{U} = \mathcal{U}_1, \ldots, \mathcal{U}_N$ register is not part of the malicious prover's state.

We therefore redefine $|\psi\rangle := |\psi\rangle_{\mathcal{Z},\mathcal{I}} |0\rangle_{\mathcal{U}}$ to denote the global state on $\mathcal{Z} \otimes \mathcal{I} \otimes \mathcal{U}$ including the ancilla $\mathcal{U}$ registers initialized to $|0\rangle_{\mathcal{U}}$.

Finally, the X_i observable is defined as

$$X_i = (U \otimes \mathsf{Id}_{\mathcal{U}})^\dagger (H^{\otimes \ell+1}_{\mathcal{Z}_i} \otimes \mathsf{Id} \otimes H_{\mathcal{U}_i}) X_i' (H^{\otimes \ell+1}_{\mathcal{Z}_i} \otimes \mathsf{Id} \otimes H_{\mathcal{U}_i})(U \otimes \mathsf{Id}_{\mathcal{U}}).$$

where

$$\begin{aligned} X_i' = & \sum_{d \in \mathsf{Good}(x_{0,i}, x_{1,i})} (-1)^{d \cdot (1, x_{0,i} \oplus x_{1,i})} |d\rangle\langle d|_{\mathcal{Z}_i} \otimes \mathsf{Id}_{\mathcal{I}, \{\mathcal{Z}_j\}_{j \neq i}, \mathcal{U}} \\ & + \sum_{d \notin \mathsf{Good}(x_{0,i}, x_{1,i}), u \in \{0,1\}} (-1)^u |d, u\rangle\langle d, u|_{\mathcal{Z}_i, \mathcal{U}_i} \otimes \mathsf{Id}_{\mathcal{I}, \{\mathcal{Z}_j\}_{j \neq i}, \{\mathcal{U}_j\}_{j \neq i}}. \end{aligned}$$

Note that the X_i observables are defined so that each pair of X_i, X_j commute. Moreover, one can verify that measuring X_i for each $i \in R$ exactly corresponds to measuring $u \in \{0,1\}^R$ as described above.

The description of X_i depends on (y_i, sk_i) because of the appearance of $x_{i,0}, x_{i,1}$ in X_i'. Moreover, note that each X_i is efficiently computable given sk_i.

For convenience, we define a procedure $\{X_i, Z_i\}_i, |\psi\rangle_{\mathcal{Z},\mathcal{I},\mathcal{U}} \leftarrow \mathsf{Samp}$ that works as follows:

- Sample keys $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{MP.Gen}(1^\lambda, N, \mathbf{1})$.
- Run the malicious prover on PK to obtain a classical commitment string y. Let $|\psi'\rangle$ denote the prover's residual state on $\mathcal{Z} \otimes \mathcal{I}$.
- Output the observables $\{X_i, Z_i\}$ parameterized by $(\mathsf{PK}, \mathsf{SK}, y)$ and malicious prover's unitary U, along with the state $|\psi\rangle := |\psi'\rangle \otimes |0\rangle_{\mathcal{U}}$.

For the remainder of this section, we will write $D_{P^*,\text{2-to-1}}$ as a two-step sampling process:

1. Run $\{X_i, Z_i\}_i, |\psi\rangle_{\mathcal{Z},\mathcal{I},\mathcal{U}} \leftarrow \mathsf{Samp}$.
2. Starting with $|\psi\rangle$, measure each Z_i for $i \in S$ to obtain $v \in \{0,1\}^S$. Then measure each X_i for $i \in R$ to obtain $u \in \{0,1\}^R$. Output $(u, v) \in \{0,1\}^R \times \{0,1\}^S$.

6.3 The Extracted State

Recall that our definition of measurement protocol soundness (Definition 4.5) requires us to give an extractor that:

1. Generates keys $(\mathsf{PK}, \mathsf{SK})$ according to an algorithm $\mathsf{SimGen}(1^\lambda)$ (independently of the verifier's basis choice h).

2. Runs the malicious prover P^* on PK to obtain y; as usual, $|\psi\rangle \in \mathcal{Z} \otimes \mathcal{I} \otimes \mathcal{U}$ denotes the residual prover state with $\mathcal{U}$ initialized to $|0\rangle_{\mathcal{U}}$.[20]

3. Generates an extracted state $\boldsymbol{\tau} \leftarrow \mathsf{Ext}^{U,|\psi\rangle}(\mathsf{PK}, \mathsf{SK}, y)$ (the superscript denotes black-box access to a unitary U and state $|\psi\rangle$, see Section 3.2).

We define $\mathsf{SimGen}(1^\lambda)$ to be $\mathsf{MP.Gen}(1^\lambda, N, \mathbf{1})$, which exactly corresponds to how keys are sampled in $D_{P^*,\text{2-to-1}}$.

To establish soundness, it remains to (1) describe how to generate the extracted state $\boldsymbol{\tau}$ given $(\mathsf{PK}, \mathsf{SK}, y), U$, and (2) prove that the distribution that arises from measuring $\boldsymbol{\tau}$ with the Pauli-X and Pauli-Z observables in the verifier's chosen bases h is computationally indistinguishable from $D_{P^*,\text{2-to-1}}$.

We handle (1) in Section 6.3.1. We then describe the distribution $D_{P^*,\mathrm{Ext}}$ that arises from measuring our extracted state in Section 6.3.2 and prove that indistinguishability from $D_{P^*,\text{2-to-1}}$ in Section 6.4.

6.3.1 A Teleportation-Inspired Extraction Procedure

Fix a choice of $\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp}$. For ease of notation, write $\mathcal{H} = \mathcal{Z} \otimes \mathcal{I} \otimes \mathcal{U}$ so that $|\psi\rangle \in \mathcal{H}$. We would like an efficient extraction procedure that takes as input $|\psi\rangle \in \mathcal{H}$ and generates an N-qubit state $\boldsymbol{\tau}$ such that, roughly speaking, measuring $|\psi\rangle$ with X/Z and measuring $\boldsymbol{\tau}$ with σ_X/σ_Z produce indistinguishable outcomes.

Intuition for the Extractor. Before we describe our extractor, we first provide some underlying intuition. For an arbitrary N-qubit Hilbert space, let $\sigma_{x,i}/\sigma_{z,i}$ denote the Pauli σ_x/σ_z observable acting on the ith qubit. For each $r, s \in \{0,1\}^N$, define the N-qubit Pauli "parity" observables

$$\sigma_x(r) := \prod_{i:r_i=1} \sigma_{x,i} \ , \ \sigma_z(s) := \prod_{i:s_i=1} \sigma_{z,i}.$$

Suppose for a moment that $|\psi\rangle \in \mathcal{H}$ is *already* an N-qubit state (i.e., $\mathcal{H}$ is an N-qubit Hilbert space) and moreover, that each X_i/Z_i observable is simply the corresponding Pauli observable $\sigma_{x,i}/\sigma_{z,i}$. While these assumptions technically trivialize the task (the state already has the form we want from the extracted state), it will be instructive to **write down an extractor that "teleports" this state into another N-qubit external register.**

[20] To match the syntax of our definition in Definition 4.5, the register $\mathcal{U}$ should be viewed as an internal register initialized by the extractor.

We can do this by initializing two N-qubit registers $\mathcal{A}_1 \otimes \mathcal{A}_2$ to $|\phi^+\rangle^{\otimes N}$ where $|\phi^+\rangle$ is the EPR state $(|00\rangle + |11\rangle)/\sqrt{2}$ (the ith EPR pair lives on the ith qubit of $\mathcal{A}_1$ and $\mathcal{A}_2$). Now consider the following steps, which are inspired by the (N-qubit) quantum teleportation protocol

1. Initialize a $2N$-qubit ancilla $\mathcal{W}$ to $\left|0^{2N}\right\rangle$, and apply $H^{\otimes 2N}$ to obtain the uniform superposition.
2. Apply a "controlled-Pauli" unitary, which does the following for all $r, s \in \{0,1\}^N$ and all $|\phi\rangle \in \mathcal{H} \otimes \mathcal{A}_1$:
$$|r,s\rangle_{\mathcal{W}} |\phi\rangle_{\mathcal{H},\mathcal{A}_1} \to |r,s\rangle_{\mathcal{W}} \left(\sigma_x(r)\sigma_z(s)_{\mathcal{H}} \otimes \sigma_x(r)\sigma_z(s)_{\mathcal{A}_1}\right) |\phi\rangle_{\mathcal{H},\mathcal{A}_1}$$
3. Apply the unitary that XORs onto $\mathcal{W}$ the outcome of performing N Bell-basis measurements[21] on $\mathcal{A}_1 \otimes \mathcal{A}_2$ onto $\mathcal{W}$, i.e., for all $u, v, r, s \in \{0,1\}^N$:
$$|u,v\rangle_{\mathcal{W}} \left(\sigma_x(r)\sigma_z(s) \otimes \mathsf{Id}\right)_{\mathcal{A}_1,\mathcal{A}_2} \left|\phi^+\right\rangle^{\otimes N}_{\mathcal{A}_1,\mathcal{A}_2} \to |u \oplus r, v \oplus s\rangle_{\mathcal{W}} \left(\sigma_x(r)\sigma_z(s) \otimes \mathsf{Id}\right)_{\mathcal{A}_1,\mathcal{A}_2} \left|\phi^+\right\rangle^{\otimes N}_{\mathcal{A}_1,\mathcal{A}_2}.$$
Finally, discard $\mathcal{W}$.

One can show that the resulting state is

$$\frac{1}{2^N} \sum_{r,s \in \{0,1\}^N} \left(\sigma_x(r)\sigma_z(s) \otimes \sigma_x(r)\sigma_z(s) \otimes \mathsf{Id}\right) |\psi\rangle_{\mathcal{H}} \left|\phi^+\right\rangle_{\mathcal{A}_1,\mathcal{A}_2} = \left|\phi^+\right\rangle_{\mathcal{H},\mathcal{A}_1} |\psi\rangle_{\mathcal{A}_2}, \tag{2}$$

where $|\psi\rangle$ is now "teleported" into the $\mathcal{A}_2$ register.

To generalize this idea to the setting where $|\psi\rangle \in \mathcal{H}$ is an arbitrary quantum state and $\{X_i, Z_i\}_i$ are an arbitrary collection of $2N$ observables, *we simply replace each $\sigma_x(r)$ and $\sigma_z(s)$ acting on $\mathcal{H}$ above with the corresponding parity observables for $\{X_i, Z_i\}$*. That is for each $r, s \in \{0,1\}^N$, define

$$Z(s) = \prod_{i=1}^{N} Z_i^{s_i} \quad \text{and} \quad X(r) = \prod_{i=1}^{N} X_i^{r_i}.$$

The rough intuition is that as long as the $\{X_i\}$ and $\{Z_i\}$ observables "behave like" Pauli observables with respect to $|\psi\rangle$, the resulting procedure will "teleport" $|\psi\rangle$ into the N-qubit register $\mathcal{A}_2$.

The Full Extractor. In more detail, we have the state $|\psi\rangle_{\mathcal{H}} = |\psi\rangle_{\mathcal{Z},\mathcal{I},\mathcal{U}}$, and we initialize two N-qubit registers $\mathcal{A}_1 \otimes \mathcal{A}_2$ to $|\phi\rangle^{\otimes N}$. We run the following steps (the changes from the above procedure are highlighted in red):

1. Initialize a $2N$-qubit ancilla $\mathcal{W}$ to $\left|0^{2N}\right\rangle$, and apply $H^{\otimes 2N}$.
2. Apply a unitary that does the following for all $r, s \in \{0,1\}^N$:
$$|r,s\rangle_{\mathcal{W}} |\phi\rangle_{\mathcal{H},\mathcal{A}_1} \to |r,s\rangle_{\mathcal{W}} \left(X(r)Z(s)_{\mathcal{H}} \otimes \sigma_x(r)\sigma_z(s)_{\mathcal{A}_1}\right) |\phi\rangle_{\mathcal{H},\mathcal{A}_1}$$

[21] The Bell basis consists of the 4 states $(\sigma_x^a \sigma_z^b \otimes \mathsf{Id})\left|\phi^+\right\rangle$ for $a, b \in \{0,1\}$ on 2 qubits.

3. Apply the unitary that XORs onto $\mathcal{W}$ the outcome of performing N Bell-basis measurements on $\mathcal{A}_1 \otimes \mathcal{A}_2$ onto $\mathcal{W}$, i.e., for all $u, v, r, s \in \{0,1\}^N$:

$$|u, v\rangle_{\mathcal{W}} (\sigma_x(r)\sigma_z(s) \otimes \mathsf{Id}) \left|\phi^+\right\rangle^{\otimes N}_{\mathcal{A}_1,\mathcal{A}_2} \to |u \oplus r, v \oplus s\rangle_{\mathcal{W}} (\sigma_x(r)\sigma_z(s) \otimes \mathsf{Id}) \left|\phi^+\right\rangle^{\otimes N}_{\mathcal{A}_1,\mathcal{A}_2} .$$

   Finally, discard $\mathcal{W}$.

All of these steps can be efficiently implemented given black-box access to $\{X_i, Z_i\}_i$. The resulting state is

$$\frac{1}{2^N} \sum_{r,s\in\{0,1\}^N} X(r)Z(s) |\psi\rangle_{\mathcal{H}} \otimes \sigma_x(r)\sigma_z(s) \left|\phi^+\right\rangle^{\otimes N}_{\mathcal{A}_1,\mathcal{A}_2} ,$$

and we define the extracted state $\boldsymbol{\tau} := \mathsf{Ext}_{\{X_i\},\{Z_i\}}(|\psi\rangle)$ to be the residual state on $\mathcal{A}_2$ after tracing out $\mathcal{H}$ and $\mathcal{A}_1$.[22]

6.3.2 Measuring the Extracted State

We now consider the N-bit distribution of measurement outcomes that arise from measuring the extracted state $\boldsymbol{\tau}$ using the Pauli observables σ_x, σ_z. In particular, we consider performing the measurements according to the verifier's basis choice, so that we measure $\sigma_{z,i}$ for each $i \in S$ and $\sigma_{x,i}$ for each $i \in R$.

Formally, we define the distribution $D_{P^*,\mathrm{Ext}}$ on $\{0,1\}^N$ obtained by the following process:

- Run $\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp}$.
- Let $\boldsymbol{\tau} = \mathsf{Ext}_{\{X_i\},\{Z_i\}}(|\psi\rangle)$ be the N-qubit extracted state.
- Measure the Pauli-Z observable $\sigma_{z,i}$ for all $i \in S$, obtaining $v \in \{0,1\}^S$.
- Measure the Pauli-X observable $\sigma_{x,i}$ for all $i \in R$, obtaining $u \in \{0,1\}^R$.
- Output $(u, v) \in \{0,1\}^R \times \{0,1\}^S$.

It will be convenient to define the following projection operators. For each $u \in \{0,1\}^R$ and $v \in \{0,1\}^S$ let

$$\Pi^{\sigma_x}_u = \mathop{\mathbb{E}}_{u'\in\{0,1\}^R} (-1)^{u\cdot u'} \sigma_x(u') \qquad \text{and} \qquad \Pi^{\sigma_z}_v = \mathop{\mathbb{E}}_{v'\in\{0,1\}^S} (-1)^{v\cdot v'} \sigma_z(v') \tag{3}$$

In words, $\Pi^{\sigma_x}_u$ is the projection that corresponds to measuring $\sigma_{x,i}$ for each $i \in R$ and obtaining the string of outcomes $u \in \{0,1\}^R$, and $\Pi^{\sigma_z}_v$ is the projection that corresponds to measuring $\sigma_{z,i}$ for each $i \in S$ and obtaining the string of outcomes $v \in \{0,1\}^S$.

Then the probability $D_{P^*,\mathrm{Ext}}$ outputs any $(u, v) \in \{0,1\}^R \times \{0,1\}^S$ can be written as

$$D_{P^*,\mathrm{Ext}}(u, v) = \mathop{\mathbb{E}}_{\{X_i,Z_i\},|\psi\rangle\leftarrow\mathsf{Samp}} [\mathrm{Tr}\left(\Pi^{\sigma_x}_u \Pi^{\sigma_z}_v \boldsymbol{\tau}\right) : \boldsymbol{\tau} = \mathsf{Ext}_{\{X_i\},\{Z_i\}}(|\psi\rangle)].$$

[22] The same extracted state is defined in Vidick's lecture notes [Vid20], although the notes do not give an explicit procedure for generating it.

We define a set of analogous projection operators for the $\{X_i\}$ and $\{Z_i\}$ observables. For each $u \in \{0,1\}^R$ and $v \in \{0,1\}^S$, let

$$\Pi_u^X = \mathop{\mathbb{E}}_{u' \in \{0,1\}^R} (-1)^{u \cdot u'} X(u') \qquad \text{and} \qquad \Pi_v^Z = \mathop{\mathbb{E}}_{v' \in \{0,1\}^S} (-1)^{v \cdot v'} Z(v') \tag{4}$$

In words, Π_u^X is the projection that corresponds to measuring X_i for each $i \in R$ and obtaining the string of outcomes $u \in \{0,1\}^R$, and Π_v^Z is the projection that corresponds to measuring Z_i for each $i \in S$ and obtaining the string of outcomes $v \in \{0,1\}^S$.

With these definitions in mind, we state a claim that allows us to characterize the result of measuring the extracted state $\boldsymbol{\tau}$ with the Pauli observables.

Claim 6.4. *Fix any choice of $\{X_i, Z_i\}_{i \in [N]}$ and state $|\psi\rangle$, and let $\boldsymbol{\tau} = \mathsf{Ext}_{\{X_i\},\{Z_i\}}(|\psi\rangle)$. For all $(u,v) \in \{0,1\}^R \times \{0,1\}^S$ it holds that*

$$\mathrm{Tr}\left(\Pi_u^{\sigma_x} \Pi_v^{\sigma_z} \boldsymbol{\tau}\right) = \mathop{\mathbb{E}}_{u' \in \{0,1\}^R} \langle\psi| \Pi_v^Z Z(u') \Pi_{u' \oplus u}^X Z(u') \Pi_v^Z |\psi\rangle . \tag{5}$$

The proof of Claim 6.4 is a straightforward (but slightly tedious) computation and is deferred to Appendix B.

Importantly, Claim 6.4 gives a clear understanding of how $D_{P^*,\text{2-to-1}}$ and $D_{P^*,\mathrm{Ext}}$ relate to each other, since it allows us to view the distribution $D_{P^*,\mathrm{Ext}}$ (which arises from Pauli measurements on the extracted state $\boldsymbol{\tau}$) as the result of performing certain protocol observable measurements $\{X_i, Z_i\}$ on $|\psi\rangle$.

Recall that the distribution $D_{P^*,\text{2-to-1}}$ is the following distribution:

1. Run $\{X_i, Z_i\}_i, |\psi\rangle_{\mathcal{Z},\mathcal{I},\mathcal{U}} \leftarrow \mathsf{Samp}$.
2. Starting with $|\psi\rangle$, measure each Z_i for $i \in S$ to obtain $v \in \{0,1\}^S$. Then measure each X_i for $i \in R$ to obtain $u \in \{0,1\}^R$. Output $(u,v) \in \{0,1\}^R \times \{0,1\}^S$.

By Claim 6.4, we can write $D_{P^*,\mathrm{Ext}}$ as follows (differences from $D_{P^*,\text{2-to-1}}$ are in red):

1. Run $\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp}$.
2. Starting with $|\psi\rangle$ measure each Z_i for $i \in S$ to obtain $v \in \{0,1\}^S$. Then sample a uniformly random string $u' \leftarrow \{0,1\}^R$ and apply the unitary $Z(u')$. Finally, measure each X_i for $i \in R$ and XOR the output with u' to obtain $u \in \{0,1\}^R$. Output $(u,v) \in \{0,1\}^R \times \{0,1\}^S$.

With this key difference in mind, it remains to prove indistinguishability of these two distributions.

6.4 Indistinguishability of Measurement Outcomes

In this subsection, we complete the proof that $D_{P^*,\text{2-to-1}}$ and $D_{P^*,\mathrm{Ext}}$ are computationally indistinguishable. We first write out their probability mass functions:

- $D_{P^*,\text{2-to-1}}$ outputs $(u,v) \in \{0,1\}^R \times \{0,1\}^S$ with probability

$$D_{P^*,\text{2-to-1}}(u,v) = \mathop{\mathbb{E}}_{\{X_i,Z_i\},|\psi\rangle \leftarrow \mathsf{Samp}} \left[\langle\psi| \Pi_v^Z \Pi_u^X \Pi_v^Z |\psi\rangle\right] .$$

- $D_{P^*,\mathrm{Ext}}$ outputs $(u,v) \in \{0,1\}^R \times \{0,1\}^S$ with probability

$$D_{P^*,\mathrm{Ext}}(u,v) = \mathop{\mathbb{E}}_{\substack{\{X_i,Z_i\},|\psi\rangle\leftarrow\mathsf{Samp}\\ u'\in\{0,1\}^R}} \left[\langle\psi|\, \Pi^Z_v Z(u')\Pi^X_{u'\oplus u} Z(u')\Pi^Z_v \,|\psi\rangle\right].$$

At this point, the reader may find it helpful to convince themselves that probability mass functions above exactly correspond to the descriptions of these distributions given at the end of Section 6.3. The equivalence between these two representations will be a key component of the upcoming proofs.

For convenience, we will reorder the indices so that the indices in R are labeled $1, 2, \ldots, |R|$. Let $u_{\le j} \in \{0,1\}^R$ be the vector equal to u on the first j indices, and is 0 on the remaining indices. For each $j \in \{0, 1, \ldots, |R|\}$, define hybrid Hyb_j to be the distribution that outputs $(u,v) \in \{0,1\}^R \times \{0,1\}^S$ with probability

$$\mathsf{Hyb}_j(u,v) = \mathop{\mathbb{E}}_{\substack{\{X_i,Z_i\},|\psi\rangle\leftarrow\mathsf{Samp}\\ u'\in\{0,1\}^R}} \left[\langle\psi|\, \Pi^Z_v Z(u'_{\le j})\Pi^X_{u\oplus u'_{\le j}} Z(u'_{\le j})\Pi^Z_v \,|\psi\rangle\right].$$

Additionally, for each $j \in \{1, \ldots, |R|\}$, and $b \in \{0,1\}$ define hybrid $\mathsf{Hyb}_{j,b}$ to be the distribution that outputs $(u,v) \in \{0,1\}^R \times \{0,1\}^S$ with probability

$$\mathsf{Hyb}_{j,b}(u,v) = \mathop{\mathbb{E}}_{\substack{\{X_i,Z_i\},|\psi\rangle\leftarrow\mathsf{Samp}\\ u'\in\{0,1\}^R}} \left[\langle\psi|\, \Pi^Z_v Z(u'_{\le j-1}) Z^b_j \Pi^X_{u\oplus u'_{\le j-1}\oplus b\cdot e_j} Z^b_j Z(u'_{\le j-1})\Pi^Z_v \,|\psi\rangle\right],$$

where $e_j \in \{0,1\}^R$ denotes the jth standard basis vector.

Claim 6.5. *For all $j \in \{1, \ldots, |R|\}$, the distributions $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ are computationally indistinguishable.*

Observe that for $j \in \{1, 2, \ldots, |R|\}$, $\mathsf{Hyb}_{j,0} = \mathsf{Hyb}_{j-1}$, and that $\mathsf{Hyb}_{j,0}$ is the uniform mixture of $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$. Since $\mathsf{Hyb}_0 = D_{P^*,\text{2-to-1}}$ and $\mathsf{Hyb}_{|R|} = D_{P^*,\mathrm{Ext}}$, Claim 6.5 implies that $D_{P^*,\text{2-to-1}}$ and $D_{P^*,\mathrm{Ext}}$ are computationally indistinguishable.

We now prove Claim 6.5, which will complete the proof of measurement protocol soundness. Our proof involves the following steps:

- First, we prove Claim 6.6, which states that the marginal distributions of $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ on $N \setminus \{j\}$ are indistinguishable due to the collapsing property of f_{pk_j}.
- We then state Claim 6.7, which (together with Claim 6.6) shows that if $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ are efficiently distinguishable, then they can be distinguished as follows:
  1. Given a sample x (from either $\mathsf{Hyb}_{j,0}$ or $\mathsf{Hyb}_{j,1}$) run an efficient algorithm A on $x_{\setminus\{j\}}$ (x without the jth bit).
  2. If A outputs 0, guess a random bit b. If A outputs 1, guess $b = x_j$.

  Roughly speaking, this reduces the task to arguing about the indistinguishability of the single bit x_j (conditioned on A outputting 1).

- Finally, we show that the 1-bit conditional distributions must be indistinguishable by appealing to the adaptive hardcore bit property of f_{pk_j}.

Claim 6.6. *Let* $R' = R \setminus \{j\}$ *and let* $(\mathsf{Hyb}_{j,0})_{[N]\setminus\{j\}}$ *and* $(\mathsf{Hyb}_{j,1})_{[N]\setminus\{j\}}$ *be the marginal distributions of* $\mathsf{Hyb}_{j,0}$ *and* $\mathsf{Hyb}_{j,1}$ *on* $[N]\setminus\{j\} = R' \cup S$*. Then* $(\mathsf{Hyb}_{j,0})_{[N]\setminus\{j\}}$ *and* $(\mathsf{Hyb}_{j,1})_{[N]\setminus\{j\}}$ *are computationally indistinguishable.*

Proof. Any quantum algorithm for distinguishing $(\mathsf{Hyb}_{j,0})_{[N]\setminus\{j\}}$ and $(\mathsf{Hyb}_{j,1})_{[N]\setminus\{j\}}$ can be represented as an $N-1$ qubit binary POVM $(A, \mathsf{Id} - A)$, where the distinguisher outputs 1 on x with probability $\langle x| A |x\rangle$. We show that this contradicts the collapsing property of f_{pk_j} (given $\mathsf{PK}, \{\mathsf{sk}_i\}_{i\neq j}$).

Consider the following adversary for the f_{pk_j} collapsing security game:

- Given $\mathsf{PK}, \{\mathsf{sk}_i\}_{i\neq j}$, the adversary runs the prover P^* on PK to obtain $(y, |\psi\rangle)$. Recall that $|\psi\rangle$ is guaranteed to contain a valid pre-image in register $\mathcal{Z}_j$. The adversary submits y to the collapsing game challenger.
- The challenger flips a random bit and either applies Z_j or does nothing.[23]
- Then the adversary performs the following steps:
  1. Measure Z_i for every $i \in S$ obtaining outcomes $v \in \{0,1\}^S$.
  2. Sample a random string $u' \leftarrow \{0,1\}^R$ and apply the unitary $Z(u'_{\leq j-1})$.
  3. Measure X_i for every $i \in R'$, and XOR the outcomes with $u'_{\leq j-1}$ to obtain an output string $u \in \{0,1\}^{R'}$.
  4. Finally, measure $|u, v\rangle$ with the POVM $\{A, \mathsf{Id} - A\}$, and output 1 if and only if the measurement outcome is A.

All of the adversary's steps can be efficiently performed given $(\mathsf{PK}, \{\mathsf{sk}_i\}_{i\neq j})$. Moreover, the above adversary's advantage in the collapsing game is polynomially related to the advantage the POVM $(A, \mathsf{Id} - A)$ attains in distinguishing $(\mathsf{Hyb}_{j,0})_{[N]\setminus\{j\}}$ and $(\mathsf{Hyb}_{j,1})_{[N]\setminus\{j\}}$. □

Given that the marginal distributions of $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ on $[N]\setminus\{j\}$ are computationally indistinguishable (Claim 6.6), we next invoke a general property of N-bit distributions implying that a distinguisher between $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ must be distinguishing some (efficiently computable) property of the jth bit of $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ *conditioned* on an efficiently computable property of the $[N]\setminus\{j\}$-marginal distributions.

Claim 6.7. *Let* $k = k(\lambda)$ *be a positive integer-valued function of a security parameter* λ*. Let* $\{D_{0,\lambda}\}_{\lambda\geq 1}$ *and* $\{D_{1,\lambda}\}_{\lambda\geq 1}$ *be families of distributions on* $\{0,1\}^{k+1}$ *such that the marginal*

[23] This version of the collapsing game is equivalent to the standard formulation in which the challenger either does/does not perform a measurement. This follows from the fact that measuring a qubit in the computational basis (and discarding the outcome) is *equivalent* to applying Z^b for a random $b \leftarrow \{0,1\}$. Thus, the challenger's measurement (in the $b = 1$ experiment) is equivalent to applying Z with probability $1/2$; for simplicity, our formulation has the challenger (in the $b = 1$ experiment) apply Z with probability 1, which increases the adversary's distinguishing advantage by a factor of 2.

distributions $D'_{0,\lambda}$ and $D'_{1,\lambda}$ of $D_{0,\lambda}$ and $D_{1,\lambda}$ respectively on the first k bits are computationally indistinguishable. Suppose that $D_{0,\lambda}$ and $D_{1,\lambda}$ are computationally distinguishable. *Then there is an efficiently computable binary-outcome POVM $\{M, \mathsf{Id} - M\}$ acting on k qubits such that*

$$\left| \mathop{\mathbb{E}}_{x \sim D_{0,\lambda}} (-1)^{x_{k+1}} \langle x_{\leq k} | M | x_{\leq k} \rangle - \mathop{\mathbb{E}}_{x \sim D_{1,\lambda}} (-1)^{x_{k+1}} \langle x_{\leq k} | M | x_{\leq k} \rangle \right| > \frac{1}{\mathsf{poly}(\lambda)}.$$

We defer the proof to Appendix C.

Finally, we show that the jth bit distinguisher of $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ discussed by Claim 6.7 cannot exist by the adaptive hardcore bit property of f_{pk_j} (given $\mathsf{PK}, \{\mathsf{sk}_i\}_{i \neq j}$).

Claim 6.8. *For any efficiently computable binary outcome POVM $\{M, \mathsf{Id} - M\}$,*

$$\left| \mathop{\mathbb{E}}_{(u,v) \sim \mathsf{Hyb}_{j,0}} (-1)^{u_j} \langle u_{\setminus\{j\}}, v | M | u_{\setminus\{j\}}, v \rangle - \mathop{\mathbb{E}}_{(u,v) \sim \mathsf{Hyb}_{j,1}} (-1)^{u_j} \langle u_{\setminus\{j\}}, v | M | u_{\setminus\{j\}}, v \rangle \right| = \mathsf{negl}(\lambda). \quad (6)$$

Proof. For the reader's convenience, we write out the probability mass functions of $\mathsf{Hyb}_{j,0}$ and $\mathsf{Hyb}_{j,1}$ explicitly, with the differences highlighted in red

$$\mathsf{Hyb}_{j,0}(u,v) = \mathop{\mathbb{E}}_{\substack{\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp} \\ u' \in \{0,1\}^R}} \left[\langle \psi | \Pi^Z_v Z(u'_{\leq j-1}) \Pi^X_{u \oplus u'_{\leq j-1}} Z(u'_{\leq j-1}) \Pi^Z_v |\psi\rangle \right]$$

$$\mathsf{Hyb}_{j,1}(u,v) = \mathop{\mathbb{E}}_{\substack{\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp} \\ u' \in \{0,1\}^R}} \left[\langle \psi | \Pi^Z_v Z(u'_{\leq j-1}) Z_j \Pi^X_{u \oplus u'_{\leq j-1} \oplus e_j} Z_j Z(u'_{\leq j-1}) \Pi^Z_v |\psi\rangle \right].$$

We define one more distribution (with the difference relative to $\mathsf{Hyb}_{j,0}$ highlighted in red)

$$\overline{\mathsf{Hyb}}_{j,1}(u,v) = \mathop{\mathbb{E}}_{\substack{\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp} \\ u' \in \{0,1\}^R}} \left[\langle \psi | \Pi^Z_v Z(u'_{\leq j-1}) Z_j \Pi^X_{u \oplus u'_{\leq j-1}} Z_j Z(u'_{\leq j-1}) \Pi^Z_v |\psi\rangle \right].$$

We now rewrite the left-hand-side of Eq. (6), where in the second expectation we sample from $\overline{\mathsf{Hyb}}_{j,1}$ instead of $\mathsf{Hyb}_{j,1}$. Note that these distributions are identical except that u_j is flipped, so we have

$$\begin{aligned} &\left| \mathop{\mathbb{E}}_{(u,v) \sim \mathsf{Hyb}_{j,0}} (-1)^{u_j} \langle u_{\setminus\{j\}}, v | M | u_{\setminus\{j\}}, v \rangle - \mathop{\mathbb{E}}_{(u,v) \sim \mathsf{Hyb}_{j,1}} (-1)^{u_j} \langle u_{\setminus\{j\}}, v | M | u_{\setminus\{j\}}, v \rangle \right| \\ &= \left| \mathop{\mathbb{E}}_{(u,v) \sim \mathsf{Hyb}_{j,0}} (-1)^{u_j} \langle u_{\setminus\{j\}}, v | M | u_{\setminus\{j\}}, v \rangle + \mathop{\mathbb{E}}_{(u,v) \sim \overline{\mathsf{Hyb}}_{j,1}} (-1)^{u_j} \langle u_{\setminus\{j\}}, v | M | u_{\setminus\{j\}}, v \rangle \right|. \end{aligned}$$

Dividing the right-hand-side by 2 gives an expression equal to the (absolute value of) the expectation of the output in the following process:

- Prepare $\{X_i, Z_i\}, |\psi\rangle \leftarrow \mathsf{Samp}$.
- Sample $b \leftarrow \{0,1\}$ and prepare $Z^b_j |\psi\rangle$ (the $b = 0$ case corresponds to $\mathsf{Hyb}_{j,0}$ and the $b = 1$ case corresponds to $\overline{\mathsf{Hyb}}_{j,1}$).
- Then measure Z_i for all $i \in S$ to obtain $v \in \{0,1\}^S$. Sample a random $u' \leftarrow \{0,1\}^R$ and apply $Z(u'_{\leq j-1})$, and finally measure X_i for all $i \in R$ and XOR the result with $u'_{\leq j-1}$ to obtain $u \in \{0,1\}^R$.

- Prepare the state $\left|u_{\setminus\{j\}}, v\right\rangle$ and measure it with the POVM $\{M, \mathsf{Id} - M\}$. If the output is $\mathsf{Id} - M$, stop at this point and output 0.
- Otherwise, if the output is M, output $(-1)^{u_j}$.

Notice that the second step is equivalent to measuring $|\psi\rangle$ with Z_j, since (writing $Z_j = Z_j^+ - Z_j^-$, where Z_j^+ is the projection onto the 1 eigenstate of Z_j and $Z_j^- = \mathsf{Id} - Z_j^+$ is the projection onto the -1 eigenstate of Z_j):

$$\frac{1}{2}(Z_j\,|\psi\rangle\langle\psi|\,Z_j + |\psi\rangle\langle\psi|) = Z_j^+\,|\psi\rangle\langle\psi|\,Z_j^+ + Z_j^-\,|\psi\rangle\langle\psi|\,Z_j^-.$$

It follows that

$$\Big|\mathop{\mathbb{E}}_{(u,v)\sim\mathsf{Hyb}_{j,0}}(-1)^{u_j}\left\langle u_{\setminus\{j\}}, v\right|M\left|u_{\setminus\{j\}}, v\right\rangle + \mathop{\mathbb{E}}_{(u,v)\sim\overline{\mathsf{Hyb}}_{j,1}}(-1)^{u_j}\left\langle u_{\setminus\{j\}}, v\right|M\left|u_{\setminus\{j\}}, v\right\rangle\Big|/2$$

is polynomially-related to the advantage of the following adversary for the adaptive hardcore bit game:

- Given $\mathsf{PK}, \{\mathsf{sk}_i\}_{i\neq j}$, the adversary runs the prover P^* on PK to obtain $(y, |\psi\rangle)$. Recall that $|\psi\rangle$ is guaranteed to contain a valid pre-image in register $\mathcal{Z}_j$ with probability $1 - \mathsf{negl}(\lambda)$.
- The adversary measures the register $\mathcal{Z}_j$ of $|\psi\rangle$ in the standard basis, obtaining a string (b_j, x_j). By the assumption that $|\psi\rangle$ contains valid pre-images and the fact that pk_j is in the range of $\mathsf{TCF.Gen}(1^\lambda, \text{2-to-1})$, this is equivalent to measuring the observable Z_j (which just measures b_j).
- Next, the adversary measures Z_i for all $i \in S$, obtaining a string of outcomes $v \in \{0,1\}^S$.
- Then the adversary samples random $u' \leftarrow \{0,1\}^R$ and applies the unitary $Z(u'_{\leq j-1})$ to its state.
- The adversary measures X_i for all $i \in R'$ and XORs the outcome with $u'_{\leq j-1}$, obtaining a string $u \in \{0,1\}^{R'}$.
- The adversary prepares the state $|u, v\rangle$ and measures it with $\{M, \mathsf{Id} - M\}$. Depending on the outcome, it does the following:
  - If the measurement outcome is $\mathsf{Id} - M$, it samples a uniformly random string $d_j \leftarrow \{0,1\}^{\ell+1}$ and sends (b_j, x_j, d_j) to the challenger (in this case obtaining $\mathsf{negl}(\lambda)$ advantage).
  - If the measurement outcome is M, it applies U to its state, followed by $H^{\otimes \ell+1}$ to $\mathcal{Z}_j$. It then measures $\mathcal{Z}_j$ to obtain a string $d_j \in \{0,1\}^{\ell+1}$ and sends (b_j, x_j, d_j) to the challenger. Note that the challenger's output bit (i.e., whether the adversary wins or loses) exactly corresponds to the bit u_j.

By assumption, this adversary outputs a valid pre-image (b_j, x_j) with probability $1 - \mathsf{negl}(\lambda)$. Since all of the adversary's steps are efficient given $(\mathsf{PK}, \{\mathsf{sk}_i\}_{i\neq j})$, the claim follows from the adaptive hardcore bit property of f_{pk_j}. □

This completes the proof of Claim 6.5, which in turn implies the soundness of the measurement protocol.

7 Succinct Key Generation from iO

In this section, we construct a cryptographic primitive that provides a succinct representation of N key pairs. We call this primitive a "succinct batch key generation algorithm," and provide definitions and a construction based on iO in Section 7.1. In Section 7.2, we compose our succinct key generation primitive with Mahadev randomized TCFs [Mah18] and prove that the composition satisfies the hypotheses stated in Section 5, while also having *succinct* keys $(\mathsf{PK}, \mathsf{SK})$.

7.1 Batch Key Generation: Definition and Construction

A batch key generation algorithm is an algorithm that outputs a description of many $(\mathsf{pk}, \mathsf{sk})$-pairs; a *succinct* batch key generation algorithm produces a short such description. Formally, we will define this primitive relative to any dual-mode key generation algorithm.

Definition 7.1. *An algorithm* Gen *is said to be a dual-mode key generation algorithm if it takes as input a security parameter* 1^λ *and a bit* $\mathsf{mode} \in \{0, 1\}$*, and it outputs a pair of keys* $(\mathsf{pk}, \mathsf{sk})$*. Moreover, we require* key indistinguishability*: public keys sampled using* $\mathsf{Gen}(1^\lambda, 0)$ *are computationally indistinguishable from public keys sampled using* $\mathsf{Gen}(1^\lambda, 1)$*.*

Definition 7.2. *Let* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode})$ *denote a dual-mode key generation algorithm. A* (succinct) batch key generation algorithm $\mathsf{BatchGen}$ *for* Gen *is a tuple of p.p.t. algorithms* $(\mathsf{Setup}, \mathsf{ExtPk}, \mathsf{ExtSk}, \mathsf{Program})$ *with the following syntax.*

- $\mathsf{Setup}(1^\lambda, N, f)$ *takes as input a security parameter* λ *in unary; the number of indices* N *in binary; and the description of a circuit* $f : [N] \to \{0, 1\}$*. It outputs a master public key* PK *and a master secret key* SK*.*
- $\mathsf{ExtPk}(\mathsf{PK}, i)$ *is a deterministic algorithm that takes as input a master public key* PK *and an index* $i \in [N]$*. It outputs a public key* pk_i*.*
- $\mathsf{ExtSk}(\mathsf{SK}, i)$ *is a deterministic algorithm takes as input a master secret key* SK *and an index* $i \in [N]$*. It outputs a secret key* sk_i*.*
- $\mathsf{Program}(1^\lambda, N, f, i, \mathsf{pk})$ *takes as input* $(1^\lambda, N, f)$ *just as* Setup *does, along with two additional inputs: an index* $i \in [N]$ *and a public key* pk*. It outputs a master public key* PK *and (an implicitly restricted) master secret key* SK*.*

We require that the following three properties are satisfied. Informally, we require that (0) $\mathsf{Setup}(1^\lambda, N, f)$ *always outputs a representation of valid key pairs, (1)* $\mathsf{Program}(1^\lambda, N, f, i, \mathsf{pk})$ *successfully programs* pk *into the* i*th "slot" of* PK*, (2) if* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode} = f(i))$ *this programming is undetectable (even given all secret keys), and (3) mode indistinguishability continues to hold for batched keys, even in the presence of "irrelevant secret keys."*

1. ***Setup Correctness.*** *For any* $\lambda, N \in \mathbb{N}$, *any circuit* $f : [N] \to \{0,1\}$, *any index* $i \in [N]$, *we have that for* $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, N, f)$ *and* $(\mathsf{pk}_i, \mathsf{sk}_i) = (\mathsf{ExtPk}(\mathsf{PK}, i), \mathsf{ExtSk}(\mathsf{SK}, i))$, $(\mathsf{pk}_i, \mathsf{sk}_i)$ *is in the range of* $\mathsf{Gen}(1^\lambda, \mathsf{mode} = f(i))$.

2. **Programming Correctness.** *For any* $\lambda, N \in \mathbb{N}$, *any circuit* $f : [N] \to \{0,1\}$, *any index* $i \in [N]$, *and any bit* mode, *we have the following guarantee: for* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode})$ *and* $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Program}(1^\lambda, N, f, i, \mathsf{pk})$,
$$\mathsf{ExtPk}(\mathsf{PK}, i) = \mathsf{pk}.$$
with probability 1.

3. **Programming Indistinguishability.** *For any* $N = N(\lambda)$, *any circuit* $f : [N] \to \{0,1\}$ *and any index* $i \in [N]$, *the following distributions are* $(\mathsf{poly}(\lambda, N), \mathsf{negl}(\lambda, N))$*-indistinguishable:*
$$\left\{(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, N, f), \mathsf{sk}_j \leftarrow \mathsf{ExtSk}(\mathsf{SK}, j) : (\mathsf{PK}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)\right\}_{\lambda \in \mathbb{N}}$$
$$\approx_c \Big\{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode} = f(i)), (\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Program}(1^\lambda, N, f, \mathsf{pk}, i),$$
$$\mathsf{sk}_i = \mathsf{sk} \text{ and } \forall j \neq i, \mathsf{sk}_j \leftarrow \mathsf{ExtSk}(\mathsf{SK}, j) : (\mathsf{PK}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)\Big\}_{\lambda \in \mathbb{N}}$$
While we let the circuit be arbitrary in this definition, we note that it will be instantiated with an efficient circuit of size $\mathsf{poly}(\log N, \lambda)$ *in our eventual constructions.*

4. **Key Indistinguishability:** *For any* $N = N(\lambda)$, *for any subset* $S \subset [N]$, *and for any two circuits* $f_0, f_1 : [N] \to \{0,1\}$ *such that* $f_0(i) = f_1(i)$ *for all* $i \in S$, *for* $(\mathsf{PK}_b, \mathsf{SK}_b) \leftarrow \mathsf{Setup}(1^\lambda, N, f_b)$, *the distributions of keys*
$$\left\{\mathsf{PK}_b, \Big(\mathsf{sk}_i \leftarrow \mathsf{ExtSk}(\mathsf{SK}_b, i)\Big)_{i \in S}\right\}_{\lambda \in \mathbb{N}}$$
are computationally $(\mathsf{poly}(\lambda, N), \mathsf{negl}(\lambda, N))$*-indistinguishable.*

We now construct succinct key generation from iO and puncturable PRFs using standard puncturing techniques.

Theorem 7.3. *For any* $N(\lambda)$, *assuming a* $(\mathsf{poly}(\lambda, N), \mathsf{negl}(\lambda, N))$*-secure iO scheme and a* $(\mathsf{poly}(\lambda, N), \mathsf{negl}(\lambda, N))$*-secure puncturable PRF, there exists a* succinct *batch key generation algorithm*
$$(\mathsf{Setup}, \mathsf{ExtPk}, \mathsf{ExtSk}, \mathsf{Program})$$
where Setup *supports batch sizes up to* $N(\lambda)$ *and runs in time* $\mathsf{poly}(\lambda, \log N)$.

In particular, when $N(\lambda) = 2^\lambda$ *we rely on the sub-exponential hardness of* $i\mathcal{O}$ *and puncturable PRFs, while for any* $N(\lambda) = \mathsf{poly}(\lambda)$ *we rely on polynomial hardness.*

Proof. Given a dual-mode key generation algorithm Gen, an iO scheme $i\mathcal{O}$, and a puncturable PRF family PRF, we define our batch key generation procedure $\mathsf{SuccGen} = (\mathsf{Setup}, \mathsf{ExtPk}, \mathsf{ExtSk}, \mathsf{Program})$ as follows.

- $\mathsf{Setup}(1^\lambda, N, f)$ samples a PRF seed s and outputs (as the public key) an obfuscated program $\widetilde{P} = \mathsf{i}\mathcal{O}(P_{s,f})$, where P is defined in Fig. 1, and (as the secret key) the PRF seed s and the function f.
- $\mathsf{ExtPk}(\mathsf{PK}, i)$ computes and outputs $\mathsf{pk}_i = \widetilde{P}(i)$ (for $\widetilde{P} = \mathsf{PK}$).
- $\mathsf{ExtSk}(\mathsf{SK}, i)$ computes $r = \mathsf{PRF}_s(i)$ and $\mathsf{mode} = f(i)$. It then computes $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode}; r)$ and outputs sk_i.
- $\mathsf{Program}(1^\lambda, N, f, \mathsf{pk}, i^*)$ samples a PRF seed s and outputs (as the public key) an obfuscated program $\mathsf{i}\mathcal{O}(P_{\mathsf{pk},i^*,s,f})$, where $P_{\mathsf{pk},i^*,s,f}$ is defined in Fig. 2, and (as the secret key) the PRF seed s and the function f.

```
Input: index $i \leq N$
Hardwired Values: Puncturable PRF seed $s$. Circuit $f$.

  • Compute $\mathsf{mode} = f(i)$ and $r = \mathsf{PRF}_s(i)$.
  • Compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode}; r)$.
  • Output $\mathsf{pk}_i$.
```

Figure 1: The program P.

```
Input: index $i \leq N$
Hardwired Values: Puncturable PRF seed $s$. Public key $\mathsf{pk}$. Index $i^*$. Circuit $f$.

  • If $i = i^*$, output $\mathsf{pk}$ and terminate.
  • Compute $\mathsf{mode} = f(i)$ and $r = \mathsf{PRF}_s(i)$.
  • Compute $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode}; r)$.
  • Output $\mathsf{pk}_i$.
```

Figure 2: The program $P_{\mathsf{pk},i^*,s,f}$.

Succinctness, setup correctness and programming correctness are immediate from the definitions. We now prove programming indistinguishability.

Claim 7.4. *For any circuit f and any index $i \in [N]$, the following distributions are* $(\mathsf{poly}(\lambda, N), \mathsf{negl}(\lambda, N))$*- computationally indistinguishable:*

$$\left\{(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, N, f), \mathsf{sk}_j \leftarrow \mathsf{ExtSk}(\mathsf{SK}, j) : (\mathsf{PK}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)\right\}$$

$$\approx_c \left\{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode} = f(i)), (\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Program}(1^\lambda, N, f, \mathsf{pk}, i),\right.$$

$$\left.\mathsf{sk}_j \leftarrow \mathsf{ExtSk}(\mathsf{SK}, j)(j \neq i), \mathsf{sk}_i = \mathsf{sk} : (\mathsf{PK}, \mathsf{sk}_1, \ldots, \mathsf{sk}_N)\right\}$$

Proof. We know that $(\mathsf{i}\mathcal{O}(P_{s,f}), s) \approx_c (\mathsf{i}\mathcal{O}(P_{\mathsf{pk},i^*,s,f}), s)$ for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode} = f(i^*); \mathsf{PRF}_s(i^*))$ by iO security because these two circuits $P_{s,f}, P_{\mathsf{pk},i^*,s,f}$ are functionally equivalent.

Moreover, $(\mathsf{i}\mathcal{O}(P_{\mathsf{pk},i^*,s,f}), \{\mathsf{sk}_i\}_{1\leq i\leq N})$ for pseudorandom $(\mathsf{pk}, \mathsf{sk}_{i^*})$ is computationally indistinguishable from $(\mathsf{i}\mathcal{O}(P_{\mathsf{pk},i^*,s,f}), \{\mathsf{sk}_i\}_{1\leq i\leq N})$ for truly random $(\mathsf{pk}, \mathsf{sk}_{i^*})$ by puncturing s at i^* (invoking iO security to do so) and then invoking PRF security. □

Finally, we prove key indistinguishability.

Claim 7.5. *For any $N = N(\lambda)$, for any subset $S \subset [N(\lambda)]$, and for any two circuits $f_0, f_1 : [N] \to \{0, 1\}$ such that $f_0(i) = f_1(i)$ for all $i \in S$, for $(\mathsf{PK}_b, \mathsf{SK}_b) \leftarrow \mathsf{Setup}(1^\lambda, N, f_b)$, the distributions of keys*

$$\left\{\mathsf{PK}_b, \left(\mathsf{sk}_i \leftarrow \mathsf{ExtSk}(\mathsf{SK}_b, i)\right)_{i\in S}\right\}_{\lambda\in\mathbb{N}}$$

are computationally $(\mathsf{poly}(\lambda, N), \mathsf{negl}(\lambda, N))$*- indistinguishable.*

Proof. Consider the following hybrid circuits f'_j for $0 \leq j \leq N$:

$$f'_j(i) = f_0(i) \text{ if } i \geq j \text{ and } f'_j(i) = f_1(i) \text{ if } i > j.$$

Note that $f'_0 = f_0$ and $f'_N = f_1$. Now, we consider the $N + 1$ distributions

$$\mathsf{Hyb}_j = \left\{\mathsf{PK}, \left(\mathsf{sk}_i \leftarrow \mathsf{ExtSk}(\mathsf{SK}, i)\right)_{i\in S}\right\}_{\lambda\in\mathbb{N}}$$

for $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, N, f'_j)$. The claim holds as long as $\mathsf{Hyb}_{j-1} \approx_c \mathsf{Hyb}_j$ for all $j \geq 1$. To see that this indistinguishability holds, it suffices to consider two further hybrid distributions:

$$\mathsf{Hyb}_{j,1} = \left\{\mathsf{PK}, \left(\mathsf{sk}_i \leftarrow \mathsf{ExtSk}(\mathsf{SK}, i)\right)_{i\neq j\in S}, \mathsf{sk}_j \text{ (included if } j \in S)\right\}_{\lambda\in\mathbb{N}}$$

for $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode} = f_0(j))$ and $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Program}(1^\lambda, N, f'_{j-1}, j, \mathsf{pk}_j)$, and

$$\mathsf{Hyb}_{j,2} = \left\{\mathsf{PK}, \left(\mathsf{sk}_i \leftarrow \mathsf{ExtSk}(\mathsf{SK}, i)\right)_{i\neq j\in S}, \mathsf{sk}_j \text{ (included if } j \in S)\right\}_{\lambda\in\mathbb{N}}$$

for $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode} = f_1(j))$ and $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Program}(1^\lambda, N, f'_j, j, \mathsf{pk}_j)$.

We have that $\mathsf{Hyb}_{j-1} \approx_c \mathsf{Hyb}_{j,1}$ by programming indistinguishability (Claim 7.4). We have that $\mathsf{Hyb}_{j,1} \approx_c \mathsf{Hyb}_{j,2}$ by considering two cases: if $f_0(j) = f_1(j)$ then $(\mathsf{pk}_j, \mathsf{sk}_j)$ are sampled from identical distributions in the hybrid and $f'_{j-1} = f'_j$, so indistinguishability follows from a single invocation of $\mathsf{i}\mathcal{O}$ security. If $f_0(j) \neq f_1(j)$, then sk_j is not included in the hybrid distributions; moreover, pk_j

in $\mathsf{Hyb}_{j,1}$ is computationally indistinguishable from pk_j in $\mathsf{Hyb}_{j,2}$ by the key indistinguishability of Gen. Finally, note that for a fixed pk_j, the programs $P_{\mathsf{pk}_j,j,s,f'_{j-1}}$ and $P_{\mathsf{pk}_j,j,s,f'_j}$ are functionally equivalent (as index j is being programmed to pk_j in both cases), so the claimed indistinguishability now follows from $\mathsf{i}\mathcal{O}$ security.

Finally, we have $\mathsf{Hyb}_{j,2} \approx_c \mathsf{Hyb}_j$ by programming indistinguishability. This completes the proof of the claim. □

This completes the proof that (Setup, ExtPk, ExtSk, Program) is a succinct batch key generation algorithm for Gen. □

7.2 Combining Succinct Key Generation with Mahadev rTCFs

In our protocols, we compose a batch key generation algorithm (Definition 7.2) with a family of Mahadev randomized TCFs (Definition 3.3). The composition is simple: use a batch key generation procedure BatchGen = (Setup, ExtPk, ExtSk, Program) to batch the procedure $\mathsf{TCF.Gen}(1^\lambda, \mathsf{mode})$ for many Mahadev rTCFs $f_{\mathsf{pk}_1}, \ldots, f_{\mathsf{pk}_N}$. The composition has the following syntax:

- $\mathsf{Setup}(1^\lambda, N, C)$ takes as input the security parameter λ, the batch size N (in binary), and a circuit C computing a function mapping $[N] \to \{0,1\}$. It outputs a public key PK and secret key SK.
- $\mathsf{ExtPk}(\mathsf{PK}, i)$ then outputs a public key pk_i that can be used to evaluate a randomized TCF f_{pk_i}.
- $\mathsf{ExtSk}(\mathsf{SK}, i)$ outputs a secret key sk_i that can be used to invert a TCF evaluation y_i.
- Program, as defined above, can be used to program a fresh $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{mode} = C(i))$ into a succinct program generated using circuit C. Program is an auxiliary algorithm used only for analysis.

We now establish that all of the necessary properties listed in Section 5 are satisfied by this composition.

Correctness of the composition (i.e., that key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ are in the range of $\mathsf{TCF.Gen}(1^\lambda, \mathsf{mode} = f(i))$) follows immediately from the correctness of Setup. Key indistinguishability of the composition is also inherited directly from the key indistinguishability of BatchGen.

We next prove that collapsing of f_{pk_j} holds in the presence of PK and all $\{\mathsf{sk}_i\}_{i \neq j}$.

Lemma 7.6. *[Collapsing] For any circuit C, any index j, and* $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, N, C)$*, the TCF f_{pk_j} is collapsing, even to an adversary given* PK along with *all secret keys $\{\mathsf{sk}_i\}_{i \neq j}$ besides sk_j.*

Formally, a computationally bounded adversary cannot win the following distinguishing game with non-negligible advantage:

1. *The adversary chooses an index $j \in [N]$ and a circuit $C : [N] \to \{0,1\}$.*
2. *The challenger samples* $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, N, C)$.

3. *The challenger sends* $(\mathsf{PK}, \{\mathsf{sk}_i\}_{i \neq j})$ *to the adversary.*
4. *The adversary prepares a quantum state* $|\psi\rangle$ *on registers* $\mathcal{B}, \mathcal{X}$ *along with a string* y *and sends both to the challenger.*
5. *The challenger computes, in superposition, whether* $\mathsf{Check}(\mathsf{pk}_j, b, x, y) = 1$*.*
    - *If* Check *fails, the challenger samples a random bit* c *and stops.*
    - *If* Check *passes, the challenger samples a random bit* c*; if* $c = 1$*, the challenger measures* $\mathcal{B}$*.*
6. *The adversary, given access to the modified* $(\mathcal{B}, \mathcal{X})$*, outputs a bit* c' *and wins if* $c' = c$*.*

Proof. We consider the following hybrid experiments.

- Hyb_0: this is the actual security game.
- Hyb_1: In step (2), challenger samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C(j))$ and samples $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Program}(1^\lambda, N, C, \mathsf{pk}, j)$.

  Hyb_0 and Hyb_1 are computationally indistinguishable by the programming indistinguishability of SuccGen.
- Hyb_2: In step (2), the challenger instead samples $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathsf{injective})$.

  Hyb_1 and Hyb_2 are computationally indistinguishable by the key indistinguishability of the injective/claw-free trapdoor functions.

Finally, in Hyb_3, even a computationally unbounded adversary cannot guess the challenge bit c, as with all but negligible probability, $\mathsf{pk}_j = \mathsf{ExtPk}(\mathsf{PK}, j)$ defines an injective function (by Definition 3.3), so after verifying that $\mathsf{Check}(\mathsf{pk}_j, b, x, y) = 1$, the register $\mathcal{B}$ is already a standard basis state. This completes the proof of Lemma 7.6. □

Finally, we prove that the adaptive hardcore bit property of f_{pk_j} holds given PK and all $\{\mathsf{sk}_i\}_{i \neq j}$

Lemma 7.7. *[Adaptive Hardcore Bit] For any* j *and any circuit* C *such that* $C(j) = 1 = \text{2-to-1}$*, for* $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Setup}(1^\lambda, N, C)$*, the adaptive hardcore bit property (see Definition 3.3) holds for the function* f_{pk_j} *(with associated secret key* sk_j*), even if the adversary is given* $(\mathsf{PK}, \{\mathsf{sk}_i\}_{i \neq j})$*.*

Proof. We consider the following hybrid experiments.

- Hyb_0: this is the adaptive hardcore bit security game for $(\mathsf{pk}_j, \mathsf{sk}_j)$ as sampled above.
- Hyb_1: this is the adaptive hardcore bit security game for $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \text{2-to-1})$, $(\mathsf{PK}, \mathsf{SK}) \leftarrow \mathsf{Program}(1^\lambda, N, C, \mathsf{pk}, j)$. The adversary is additionally given PK and $\mathsf{sk}_i = \mathsf{ExtSk}(\mathsf{SK}, i)$ for all $i \neq j$.

Hyb_0 and Hyb_1 are computationally indistinguishable by the programming indistinguishability property. Moreover, the adversary's advantage in Hyb_1 is negligible by the adaptive hardcore bit property of the freshly generated key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \text{2-to-1})$, as a reduction given pk can simulate Hyb_1 by sampling all other parameters given to the Hyb_1 adversary itself.

This completes the proof of Lemma 7.7. □

8 A Verifier-Succinct Protocol

In this section, we present a delegation protocol for **QMA** with succinct verifier messages. First, in Section 8.1, we describe results due to [ACGH20] about the parallel repetition of certain commit-challenge-response protocols with a quantum prover. Our treatment is somewhat more abstract than [ACGH20], so for completeness we provide proofs of all claims (based on proofs appearing in [ACGH20]). Next, in Section 8.2, we describe the syntax of a non-interactive, information-theoretic **QMA** verification protocol (with quantum verifier) that we will use, due to [FHM18]. In Section 8.3, we describe a *verifier-succinct* protocol for **QMA** delegation, where the verifier messages (but not the prover messages) are succinct.

8.1 Quantum commit-challenge-response protocols

Consider any *commit-challenge-response* protocol between a quantum prover P and a classical verifier V, with the following three phases.

- Commit: $P(1^\lambda)$ and $V(1^\lambda; r)$ engage in a (potentially interactive) commitment protocol, where r are the random coins used by V.
- Challenge: V samples a random bit $b \leftarrow \{0,1\}$ and sends it to P.
- Response: P computes a (classical) response z and sends it to V.

After receiving the response, V decides to accept or reject the execution.

Consider any non-uniform QPT prover P^*, and let $\left|\psi_{\lambda,r}^{P^*}\right\rangle_{\mathcal{A},\mathcal{C}}$ be the (purified) state of the prover after interacting with $V(1^\lambda; r)$ in the commit phase, where $\mathcal{C}$ holds the (classical) prover messages output during this phase, and $\mathcal{A}$ holds the remaining state.

The remaining strategy of the prover can be described by family of unitaries $\left\{U_{\lambda,0}^{P^*}, U_{\lambda,1}^{P^*}\right\}_{\lambda\in\mathbb{N}^+}$, where $U_{\lambda,0}^{P^*}$ is applied to $\left|\psi_{\lambda,r}^{P^*}\right\rangle$ on challenge 0 (followed by a measurement of z), and $U_{\lambda,1}^{P^*}$ is applied to $\left|\psi_{\lambda,r}^{P^*}\right\rangle$ on challenge 1 (followed by a measurement of z).

Let $V_{\lambda,r,0}$ denote the accept/reject predicate applied by the verifier to the prover messages when $b = 0$, written as a projection to be applied to the registers holding the prover messages, and define $V_{\lambda,r,1}$ analogously. Then define the following projectors on $\mathcal{A} \otimes \mathcal{C}$:

$$\Pi_{\lambda,r,0}^{P^*} \coloneqq U_{\lambda,0}^{P^*\dagger} V_{\lambda,r,0} U_{\lambda,0}^{P^*}, \quad \Pi_{\lambda,r,1}^{P^*} \coloneqq U_{\lambda,1}^{P^*\dagger} V_{\lambda,r,1} U_{\lambda,1}^{P^*}.$$

Definition 8.1. *A* commit-challenge-response *protocol has* computationally orthogonal projectors *if for any QPT prover P^*,*

$$\mathop{\mathbb{E}}_{r}\left[\left\langle \psi_{\lambda,r}^{P^*}\right| \Pi_{\lambda,r,0}^{P^*} \Pi_{\lambda,r,1}^{P^*} \Pi_{\lambda,r,0}^{P^*} \left|\psi_{\lambda,r}^{P^*}\right\rangle\right] = \mathsf{negl}(\lambda).$$

Proofs of the following are given in Appendix A.

Lemma 8.2 ([ACGH20]). *Consider a commit-challenge-response protocol with the following properties.*

1. *$V_{\lambda,r,0}$ does not depend on r (that is, it is publicly computable given the transcript).*
2. *For any P^*, if $\mathbb{E}_r \left[\left\langle \psi^{P^*}_{\lambda,r} \middle| \Pi^{P^*}_{\lambda,r,0} \middle| \psi^{P^*}_{\lambda,r} \right\rangle \right] = 1 - \mathsf{negl}(\lambda)$, then $\mathbb{E}_r \left[\left\langle \psi^{P^*}_{\lambda,r} \middle| \Pi^{P^*}_{\lambda,r,1} \middle| \psi^{P^*}_{\lambda,r} \right\rangle \right] = \mathsf{negl}(\lambda)$.*

Then, the protocol has computationally orthogonal projectors.

Theorem 8.3 ([ACGH20]). *Consider the λ-fold parallel repetition of any commit-challenge-response protocol with computationally orthogonal projectors. The probability that the verifier accepts all λ parallel repetitions of the protocol is $\mathsf{negl}(\lambda)$.*

8.2 Non-Interactive Post Hoc Verification of QMA

We recall a useful information-theoretic QMA verification protocol of Fitzsimons, Hajdušek, and Morimae [FHM18]. In fact, we will use an "instance-independent" version due to [ACGH20].

Lemma 8.4 ([FHM18, ACGH20]). *For all languages $\mathcal{L} = (\mathcal{L}_{\mathsf{yes}}, \mathcal{L}_{\mathsf{no}}) \in$ **QMA** there exists a polynomial $k(\lambda)$, a function $\ell(\lambda)$ that is polynomial in the time $T(\lambda)$ required to verify instances of size λ, a QPT algorithm P_{FHM}, and a PPT algorithm V_{FHM} such that the following holds.*

- *$P_{\mathsf{FHM}}(x, |\psi\rangle) \to |\pi\rangle$: on input an instance $x \in \{0,1\}^\lambda$ and a quantum state $|\psi\rangle$, P_{FHM} outputs an $\ell(\lambda)$-qubit state $|\pi\rangle$.*
- ***Completeness.** For all $x \in \mathcal{L}_{\mathsf{yes}}$ and $|\phi\rangle \in \mathcal{R}_{\mathcal{L}}(x)$ it holds that*

$$\Pr\Big[V_{\mathsf{FHM}}(x, M(h, |\pi\rangle)) = \mathsf{acc} : |\pi\rangle \leftarrow P_{\mathsf{FHM}}\left(x, |\phi\rangle^{\otimes k(\lambda)}\right)\Big] \geq 1 - \mathsf{negl}(\lambda)$$

  where $h \leftarrow \{0,1\}^{\ell(\lambda)}$.

- ***Soundness.** For all $x \in \mathcal{L}_{\mathsf{no}}$ and all ℓ-qubit states $|\pi^*\rangle$ it holds that*

$$\Pr[V_{\mathsf{FHM}}(x, M(h, |\pi^*\rangle)) = \mathsf{acc}] \leq \mathsf{negl}(\lambda)$$

  where $h \leftarrow \{0,1\}^{\ell(\lambda)}$.

*Moreover, when $\mathcal{L} \in$ **BQP**, the honest prover algorithm P_{FHM} is also a **BQP** algorithm.*

While the result was originally stated in [FHM18, ACGH20] to have an inverse polynomial soundness gap, we have driven the soundness gap to negligible by standard QMA amplification. Finally, we remark that although the algorithm V_{FHM} is completely classical, the entire verification procedure is quantum since it involves measuring the quantum state sent by the prover.

8.3 Semi-Succinct Delegation for QMA

We describe a protocol for verifying any QMA language $\mathcal{L}$.

Ingredients:

- Let $(P_{\mathsf{FHM}}, V_{\mathsf{FHM}})$ be the non-interactive protocol described in Lemma 8.4 for language $\mathcal{L}$ with associated polynomials $k(\lambda), \ell(\lambda)$.
- Let $\mathsf{PRF} : \{0,1\}^\lambda \times \{0,1\}^{\log \ell(\lambda)} \to \{0,1\}$ be a pseudo-random function.
- Let $P_{\mathsf{Meas}} = (\mathsf{Commit}, \mathsf{Open})$ and $V_{\mathsf{Meas}} = (\mathsf{Gen}, \mathsf{Test}, \mathsf{Out})$ be the prover and verifier algorithms for an $\ell(\lambda)$-qubit (verifier succinct) commit-and-measure protocol, defined in Section 4 and constructed in Section 5.1.

The Protocol:

- The verifier is initialized with an instance $x \in \{0,1\}^\lambda$ and the prover is initialized with x and $k(\lambda)$ copies of a witness $|\phi\rangle \in \mathcal{R}_{\mathcal{L}}(x)$.
- The verifier samples $s \leftarrow \{0,1\}^\lambda$, defines C so that $C(i) = \mathsf{PRF}_s(i)$, and computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)$. It sends pk to the prover.
- The prover first computes $|\psi\rangle \leftarrow P_{\mathsf{FHM}}\left(x, |\phi\rangle^{\otimes k(\lambda)}\right)$, and then computes $(y, |\mathsf{st}\rangle) \leftarrow \mathsf{Commit}(\mathsf{pk}, |\psi\rangle)$. It sends y to the verifier.
- The verifier samples a random challenge $c \leftarrow \{0,1\}$ and sends c to the prover.
- The prover computes $z \leftarrow \mathsf{Open}(|\mathsf{st}\rangle, c)$ and sends z to the verifier.
- If $c = 0$, the verifier checks whether $\mathsf{Test}(\mathsf{pk}, (y,z)) = \mathsf{acc}$ and rejects if the test fails. If $c = 1$, the verifier computes $m \leftarrow \mathsf{Out}(\mathsf{sk}, (y,z))$ and checks whether $V_{\mathsf{FHM}}(x, m) = \mathsf{acc}$. The verifier accepts if and only the verification is successful.

Theorem 8.5. *Let $(P_{\mathsf{SS}}, V_{\mathsf{SS}})$ be the λ-fold parallel repetition of the above protocol. Then, $(P_{\mathsf{SS}}, V_{\mathsf{SS}})$ satisfies completeness and soundness as defined in Definition 3.1. Moreover, for an instance x with QMA verification time T, the total size of verifier messages is* $\mathsf{poly}(\lambda, \log T)$.

Proof. First, the verifier message size guarantee follows from the fact that the verifier initializes the commit-and-measure protocol with a circuit C that succinctly encodes $\ell(\lambda)$ bits using a PRF with input size $\log(\ell(\lambda))$, where $\ell(\lambda)$ is polynomially related to the QMA verification time T.

Next, we argue completeness. For any $x \in \mathcal{L}$, we show that a single repetition of the protocol accepts with $1-\mathsf{negl}(\lambda)$ probability, and so completeness of the λ-fold parallel repetition then follows by a union bound. In the case of a test round, this follows from the test round completeness of the commit-and-measure protocol (Definition 4.2). In the case of a measurement round, we first see that measurement round completeness of the commit-and-measure protocol (Definition 4.2) implies that the probability that the verifier outputs 1 is $\mathsf{negl}(\lambda)$-close to the probability that the FHM protocol $(P_{\mathsf{FHM}}, V_{\mathsf{FHM}})$ accepts when run with an honest prover, but where $h = (\mathsf{PRF}_s(1), \ldots, \mathsf{PRF}_s(\ell))$ for $s \leftarrow \{0,1\}^\lambda$. Then, the security of the PRF implies that this probability is $\mathsf{negl}(\lambda)$-close to the probability that the FHM protocol accepts when $h \leftarrow \{0,1\}^\ell$. Finally, this is $\mathsf{negl}(\lambda)$-close to 1 by the completeness of the FHM protocol (Lemma 8.4).

Finally, we argue soundness. Consider any $x \notin \mathcal{L}$. By Theorem 8.3, it suffices to show that the single repetition of the protocol satisfies the conditions of Lemma 8.2. We define $V_{\lambda,r,0}$ to be the verifier's accept projection on a test round, and $V_{\lambda,r,1}$ to be the verifier's accept projection on a measurement round. Condition 1 of Lemma 8.2 follows immediately from the structure of the commit-and-measure protocol. Now, consider any prover P^* such that the first expectation in condition 2 is $1 - \mathsf{negl}(\lambda)$, meaning that P^* passes the test round with $1 - \mathsf{negl}(\lambda)$ probability. By the soundness of the commit-and-measure protocol, there exists a state ρ such that the probability that the verifier accepts on a measurement round is $\mathsf{negl}(\lambda)$-close to the probability that V_{FHM} accepts given x, h, and $M(h, \rho)$, where $h = (\mathsf{PRF}_s(1), \ldots, \mathsf{PRF}_s(\ell))$ for $s \leftarrow \{0,1\}^\lambda$. By security of the PRF, this probability is $\mathsf{negl}(\lambda)$-close to the probability that V_{FHM} accepts given x, h, and $M(h, \rho)$, where $h \leftarrow \{0, 1\}$. Since $x \notin \mathcal{L}$, soundness of the FHM protocol implies that this is $\mathsf{negl}(\lambda)$. Thus, the second expectation in condition 2 of Lemma 8.2 is $\mathsf{negl}(\lambda)$, which establishes that this condition is satisfied, and completes the proof.

□

9 The Fully Succinct Protocol

In this section, we compile the verifier-succinct delegation scheme from Section 8 into a full-fledged delegation scheme for **QMA**. Formally, we assume the existence of a delegation scheme for **QMA** satisfying the following properties:

1. All verifier messages can be computed in time $\mathsf{poly}(\lambda, \log N)$. (This is the definition of verifier succinctness.)

2. Moreover, the verifier messages can be computed *obliviously* to the **QMA** instance and the prover messages (this holds for the Section 8 protocol).

We present two compilers enabling this:

1. The first (and simpler) compiler only additionally assumes the existence of a collapsing hash function, which is implied by LWE (Lemma 3.7). It converts a $2r$-round verifier-succinct protocol into a $4(r + 1)$ round fully succinct protocol. In particular, the protocol from Section 8 is compiled into a 12-round succinct argument for **QMA**.

2. The second compiler additionally assumes collapsing hash function *and* a (classical, post-quantum) fully homomorphic encryption (FHE) scheme. It converts a $2r$ round verifier-succinct protocol to a $4r$ round fully succinct protocol. *Moreover*, if the verifier-succinct protocol is public-coin (except for the first message), then so is the fully succinct protocol. This results in an 8 round succinct argument system for **QMA** that is public-coin except for the first message.

For simplicity, we write down the compiled protocols in the case $r = 2$, corresponding to the protocols from Section 8.

Our main tool for these compilers are post-quantum succinct arguments of knowledge for NP [CMSZ21, LMS21]. Specifically, the security guarantees proved in [CMSZ21] are insufficient for the compilers, because the post-quantum extraction algorithm from [CMSZ21] is not sufficiently *composable* since their extractor might significantly disturb the prover's state. Instead, we make use of a composable variant of the [CMSZ21] extractor due to [LMS21] called "state-preserving succinct arguments of knowledge,"[24] which we now define.

9.1 State-Preserving Succinct Arguments of Knowledge

Definition 9.1. *A publicly verifiable argument system Π for an* NP *language L (with witness relation R) is an ϵ-state-preserving succinct argument-of-knowledge if it satisfies the following properties.*

- *Succinctness: when invoked on a security parameter λ and instance size n and a relation decidable in time T, the communication complexity of the protocol is $\mathsf{poly}(\lambda, \log T)$. The verifier computational complexity is $\mathsf{poly}(\lambda, \log T) + \tilde{O}(n)$.*
- *ϵ-State-Preserving Extraction. There exists an extractor $E^{(\cdot)}(x, \epsilon)$ with the following properties*
  - *Efficiency: $E^{(\cdot)}(x, \epsilon)$ runs in time $\mathsf{poly}(n, \lambda, 1/\epsilon)$ as a quantum oracle algorithm (with the ability to apply controlled U-gates given an oracle $U(\cdot)$), outputting a classical transcript $\tilde{\tau}$ and a classical string w.*
  - *State-preserving: Let $|\psi\rangle \in \mathcal{A} \otimes \mathcal{I}$ be any $\mathsf{poly}(\lambda)$-qubit pure state and let $\boldsymbol{\rho} = \mathrm{Tr}_{\mathcal{A}}(|\psi\rangle) \in \mathrm{D}(\mathcal{I})$.[25] Consider the following two games:*
    - *Game 0 (Real) Generate a transcript τ by running $P^*(\boldsymbol{\rho}_{\mathcal{I}}, x)$ with the honest verifier V. Output τ along with the residual state on $\mathcal{A} \otimes \mathcal{I}$.*
    - *Game 1 (Simulated) Generate a transcript-witness pair $(\tilde{\tau}, w) \leftarrow E^{P^*(\boldsymbol{\rho}_{\mathcal{I}}, x)}$. Output $\tilde{\tau}$ and the residual state on $\mathcal{A} \otimes \mathcal{I}$.*

    Then, we have that the output distributions of Game 0 and Game 1 are computationally ε-indistinguishable to any quantum distinguisher.
  - *Extraction correctness: for any P^* as above, the probability that $\tilde{\tau}$ is an accepting transcript but w is* not *in R_x is at most $\epsilon + \mathsf{negl}(\lambda)$.*

Theorem 9.2 ([LMS21]). *Assuming the post-quantum $\mathsf{poly}(\lambda, 1/\epsilon)$ hardness of learning with errors, there exists a (4-message, public coin) ϵ-state preserving succinct argument of knowledge for* NP*.*

[24] We only require a weak variant of what was constructed in [LMS21], where state preservation is allowed an inverse polynomial ϵ error.

[25] In general, the prover's input state on $\mathcal{I}$ may be entangled with some external register $\mathcal{A}$, and we ask that computational indistinguishability holds even given $\mathcal{A}$. Our definition is stated this way for maximal generality, though we remark that the applications in this section do not require indistinguishability in the presence of an entangled external register.

9.2 The QMA Protocol, Version 1

Let SemiSuccinct denote a verifier-succinct **QMA** delegation scheme additionally satisfying verifier obliviousness. For simplicity, we assume that SemiSuccinct is a four-round protocol. We formalize the execution of SemiSuccinct on a **QMA** instance x as follows:

- The verifier computes and sends $\mathsf{pk} \leftarrow \mathsf{SemiSuccinct}.V_1(1^\lambda, 1^{|x|}; r)$ (obliviously to the instance x) with randomness $r \leftarrow \{0,1\}^\lambda$.
- The prover, on initial state $|\psi\rangle$, computes $(y, \boldsymbol{\rho}) \leftarrow \mathsf{SemiSuccinct}.P(1^\lambda, \mathsf{pk}, |\psi\rangle)$, which results in a message y and residual state $\boldsymbol{\rho}$.
- The verifier computes and sends $\beta = \mathsf{SemiSuccinct}.V_2(r)$, obliviously to the instance x and the prover message y.
- The prover computes and sends $z \leftarrow \mathsf{SemiSuccinct}.P(\boldsymbol{\rho}, \beta)$.
- The verifier computes and outputs a (potentially expensive) predicate $V(x, y, z, r)$.

Finally, let AoK denote the state-preserving succinct argument of knowledge of Theorem 9.2, and let H denote a collapsing hash function family mapping $\{0,1\}^*$ to $\{0,1\}^\lambda$. Our succinct **QMA** delegation protocol QMArg is defined as follows.

1. The verifier computes and sends $\mathsf{pk} = \mathsf{SemiSuccinct}.V_1(1^\lambda, 1^{|x|}; r)$ with randomness $r \leftarrow \{0,1\}^\lambda$, along with a hash function $h \leftarrow H_\lambda$.
2. The prover computes $(y, \boldsymbol{\rho}) \leftarrow \mathsf{SemiSuccinct}.P(1^\lambda, \mathsf{pk}, |\psi\rangle)$ and sends $\hat{y} = h(y)$.
3. The prover and verifier execute AoK on the statement "$\exists w$ such that $\hat{y} = h(w)$."
4. The verifier computes and sends $\beta = \mathsf{SemiSuccinct}.V_2(r)$. Note that $\mathsf{SemiSuccinct}.V_2$ is oblivious to the prover message and so can be computed without it.
5. The prover computes $z \leftarrow \mathsf{SemiSuccinct}.P(\boldsymbol{\rho}, \beta)$ and sends $\hat{z} = h(z)$.
6. The prover and verifier execute AoK on the statement "$\exists w$ such that $\hat{z} = h(w)$."
7. The verifier sends r.
8. The prover and verifier execute AoK on the statement $\exists w_1, w_2$ such that $\hat{y} = h(w_1), \hat{z} = h(w_2)$, and $V(x, w_1, w_2, r) = 1$.

Completeness of the protocol follows directly from the completeness of SemiSuccinct and AoK. Moreover, succinctness follows directly from the compression of H, the verifier succinctness of SemiSuccinct, and the succinctness of AoK.

Since AoK has a round complexity of 4 and the first message can be re-used (indeed, h can be used as the first message for AoK), the round complexity of QMArg is 12.

9.2.1 Proof of Soundness

Theorem 9.3. *Assume that* $\mathsf{SemiSuccinct}$ *is (post-quantum) computationally sound,* H *is collapsing,*[26] *and that* AoK *is an* ϵ*-state-preserving argument of knowledge. Then,* QMArg *is (post-quantum) computationally sound.*

Proof. Let $x \notin L$ and suppose that a QPT $P^*(\boldsymbol{\rho}, x)$ breaks the soundness of QMArg with probability ϵ^*. We use P^*, together with the soundness guarantees of AoK and the collision resistance property of H, to break the soundness of $\mathsf{SemiSuccinct}$. In particular, consider the following attack on the soundness of $\mathsf{SemiSuccinct}$:

- Set an accuracy parameter $\epsilon = \frac{\epsilon^*}{10}$. Whenever we call the AoK extractor E, we will use accuracy parameter ϵ.
- Given a verifier message pk, we feed (h, pk) to $P^*(\boldsymbol{\rho}, x)$ and obtain a hash value $\hat{y}$. Then, we run the AoK extractor E on P^*'s execution of step (3) (the first execution of AoK), outputting a triple $(\tilde{\tau}_1, \tilde{\boldsymbol{\rho}}_1, y)$. We send y to the verifier.
- Given the verifier challenge β, we run $P^*(\tilde{\boldsymbol{\rho}}_1, \mathsf{pk}, h, \tilde{\tau}_1)$ to obtain a message $\hat{z}$. Then, we run the AoK extractor E on P^*'s execution of step (6), obtaining a triple $(\tilde{\tau}_2, \tilde{\boldsymbol{\rho}}_2, z)$. We send z to the verifier.

Finally, to analyze the behavior of this attack, we consider the following additional step (this is only a mental experiment).

- Given the secret verifier randomness r, run the AoK extractor E on P^*'s execution of step (8) to obtain a triple $(\tilde{\tau}_3, \tilde{\boldsymbol{\rho}}_3, y', z')$.

Claim 9.4. *With probability at least* ϵ *over the attack experiment, we have that* $\mathsf{SemiSuccinct}.V(x, \mathbf{y}, \mathbf{z}, r) = 1$.

Note that this claim contradicts the soundness of $\mathsf{SemiSuccinct}$.

Proof. The equation $\mathsf{SemiSuccinct}.V(x, \mathbf{y}, \mathbf{z}, r) = 1$ follows from the following properties of an execution of the mental experiment:

- $h(y) = \hat{y}$
- $h(z) = \hat{z}$
- $h(y') = \hat{y}$
- $h(z') = \hat{z}$
- $\mathsf{SemiSuccinct}.V(x, y', z', r) = 1$.

[26] Collision-resistance of H suffices.

The above suffices because it implies that $(y,z) = (y',z')$ except with negligible probability by the collapsing (or just collision-resistance) of H, and so the last equation implies that $\mathsf{SemiSuccinct}.V(x,y,z,r) = 1$ (except with negligible probability).

Finally, we note that all five of the above conditions simultaneously hold with probability at least ϵ by the state-preservation and correctness of E. More specifically,

- The transcript $(x, \mathsf{pk}, h, \hat{y}, \tilde{\tau}_1, \beta, \hat{z}, \tilde{\tau}_2, r, \tilde{\tau}_3)$ is accepting (according to the QMArg verifier) with probability at least $\epsilon^* - 3\epsilon$. This follows by a hybrid argument invoking the state preservation of E^* on the three executions of AoK (first w.r.t. $\tilde{\tau}_1$, then $\tilde{\tau}_2$, then $\tilde{\tau}_3$).
- Then, the correctness property of E implies that all five conditions hold simultaneously with probability at least $\epsilon^* - 6\epsilon - \mathsf{negl}(\lambda)$. □

This completes the proof of soundness of QMArg. □

9.3 The QMA Protocol, Version 2

We now describe a public-coin variant of the Section 9.2 transformation that additionally uses a Fully Homomorphic Encryption (FHE) scheme $\mathsf{FHE} = (\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$.

Let SemiSuccinct and AoK denote the argument systems from Section 9.2. Then, our second succinct argument system QMArg_2 is defined as follows.

1. The verifier computes and sends $\mathsf{pk} = \mathsf{SemiSuccinct}.V_1(1^\lambda, 1^{|x|}; r)$ with randomness $r \leftarrow \{0,1\}^\lambda$, along with a hash function $h \leftarrow H_\lambda$.
2. The verifier also samples $(\mathsf{FHE.pk}, \mathsf{FHE.sk}) \leftarrow \mathsf{FHE.Gen}(1^\lambda)$ and computes FHE ciphertext $\mathsf{ct}_V = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, r)$. The verifier sends $\mathsf{FHE.pk}, \mathsf{ct}_V$ to the prover.
3. The prover computes $(y, \boldsymbol{\rho}) \leftarrow \mathsf{SemiSuccinct}.P(1^\lambda, \mathsf{pk}, |\psi\rangle)$ and sends $\hat{y} = h(y)$.
4. The prover and verifier execute AoK on the statement "$\exists w$ such that $\hat{y} = h(w)$."
5. The verifier computes and sends $\beta = \mathsf{SemiSuccinct}.V_2(r)$. Note that $\mathsf{SemiSuccinct}.V_2$ is oblivious to the prover message and so can be computed without it.
6. The prover computes $z \leftarrow \mathsf{SemiSuccinct}.P(\boldsymbol{\rho}, \beta)$ and sends $\hat{z} = h(z)$. The prover also computes $\mathsf{ct}_P = \mathsf{FHE.Eval}(V(x,y,z,\cdot), \mathsf{ct}_V)$ and sends ct_P to the verifier.
7. The prover and verifier execute AoK on the statement $\exists w_1, w_2$ such that $\hat{y} = h(w_1), \hat{z} = h(w_2)$, and $\mathsf{ct}_P = \mathsf{FHE.Eval}(V(x,y,z,\cdot), \mathsf{ct}_V)$.
8. The verifier checks that $\mathsf{FHE.Dec}(\mathsf{ct}_P) = 1$.

As before, completeness and succinctness follow immediately from the definitions. Additionally, we note that the round complexity has been reduced to 8 because AoK is only invoked twice. Finally, we note that as long as SemiSuccinct and AoK are public-coin (except for the first message of SemiSuccinct), then QMArg_2 is also public-coin (except for the first verifier message).

9.3.1 Proof of Soundness

Theorem 9.5. *Assume that* $\mathsf{SemiSuccinct}$ *is (post-quantum) computationally sound,* H *is collapsing,* FHE *is semantically secure, and that* AoK *is an* ϵ*-state-preserving argument of knowledge. Then,* QMArg_2 *is (post-quantum) computationally sound.*

Proof. Let $x \notin L$ and suppose that a QPT $P^*(\boldsymbol{\rho}, x)$ breaks the soundness of QMArg_2 with probability ϵ^*. We use P^*, together with the soundness guarantees of AoK, the semantic security of FHE, and the collision resistance property of H, to break the soundness of $\mathsf{SemiSuccinct}$. In particular, consider the following attack on the soundness of $\mathsf{SemiSuccinct}$:

- Set an accuracy parameter $\epsilon = \frac{\epsilon^*}{10}$. Whenever we call the AoK extractor E, we will use accuracy parameter ϵ.
- Given a verifier message pk, we sample $h, \mathsf{FHE.pk}$ ourselves and feed $(h, \mathsf{pk}, \mathsf{FHE.pk}, \mathsf{ct}_V = \mathsf{FHE.Enc}(\mathsf{FHE.pk}, 0))$ to $P^*(\boldsymbol{\rho}, x)$ and obtain a hash value $\hat{y}$. Then, we run the AoK extractor E on P^*'s execution of step (3) (the first execution of AoK), outputting a triple $(\tilde{\tau}_1, \tilde{\boldsymbol{\rho}}_1, y)$. We send y to the verifier.
- Given the verifier challenge β, we run $P^*(\tilde{\boldsymbol{\rho}}_1, \mathsf{pk}, h, \tilde{\tau}_1)$ to obtain a message $\hat{z}, \mathsf{ct}_P$. Then, we run the AoK extractor E on P^*'s execution of step (6), obtaining a triple $(\tilde{\tau}_2, \tilde{\boldsymbol{\rho}}_2, y', z)$. We send z to the verifier.

We claim that this attack breaks the soundness of $\mathsf{SemiSuccinct}$ – meaning that $V(x, y, z, r) = 1$ – with probability at least $\epsilon^* - 4\epsilon - \mathsf{negl}(\lambda)$. To prove this, by FHE semantic security, it suffices to show the same thing when ct_V is instead sampled as $\mathsf{FHE.Enc}(\mathsf{FHE.pk}, r)$.

From here, the proof proceeds similarly to the proof of Theorem 9.3. In particular, the equation $\mathsf{SemiSuccinct}.V(x, y, z, r) = 1$ follows from the following properties of the hybrid attack execution:

- $h(y) = \hat{y}$
- $h(y') = \hat{y}$
- $h(z) = \hat{z}$
- $\mathsf{FHE.Eval}(V(x, y', z, \cdot), \mathsf{ct}_V) = \mathsf{ct}_P$
- $\mathsf{FHE.Dec}(\mathsf{FHE.sk}, \mathsf{ct}_P) = 1$.

This suffices due to the collapsing of H and the correctness of $\mathsf{FHE.Eval}$. By the same argument as in the proof of Theorem 9.3, these properties simultaneously hold with probabiltiy at least $\epsilon^* - 4\epsilon - \mathsf{negl}(\lambda)$ by the state-preserving extraction properties of AoK.

This completes the proof of soundness of QMArg_2.

□

10 Additional Results

In this section, we describe a number of additional new results that follow from our template for building succinct arguments for QMA. First, we show how to compile the protocol from Section 9.3 into a *two-message* succinct argument for QMA in the quantum random oracle model. We sometimes refer to such argument systems as designated-verifier SNARGs (dvSNARGs) in the QROM. Next, we show how to obtain batch arguments for QMA (where the communication size only depends on a *single* instance size) from only the quantum hardness of learning with errors (i.e. without indistinguishability obfuscation). Finally, we describe how to add *zero-knowledge* to our succinct argument in the plain model and to our dvSNARG in the QROM.

10.1 Succinct Non-interactive Arguments in the QROM

Consider any constant-round protocol (P, V) for language $\mathcal{L}$ that is public-coin except for the first message. That is, the verifier is defined by two circuits (V_0, V_1). Given instance x, the verifier first samples random coins r and computes a first message $s_0 = V_0(x, r)$. Then, the subsequent verifier message are uniformly random strings $s_1, \dots, s_c$ of at least λ bits. Finally, the verifier computes a circuit $V_1(x, r, s_0, t_0, s_1, t_1, \dots, s_c, t_c)$ that determines whether it accepts or rejects, where $t_0, \dots, t_c$ are the prover messages. Let $H : ((I \times S) \cup S) \times ([c] \times T) \to S$ be a random oracle, where I is the space of instances, S is the space of verifier messages, and T is the space of prover messages. Let $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ be the following protocol.

- Given x, V_{FS} samples r and outputs $s_0 = V_0(x, r)$.
- P_{FS} runs P on (x, s_0) to obtain t_0. Then it computes $s_1 = H((x, s_0), (0, t_0))$ and continues to run P on s_1 to obtain t_1. Then for $i \in [c]$, it computes $s_i = H(s_{i-1}, (i, t_{i-1}))$ and continues to run P on s_i to obtain t_i. Finally, it sends $(t_0, \dots, t_c)$.
- V_{FS} checks that $s_1 = H((x, s_0), (0, t_0))$ and that for each $i \in [2, \dots, c]$, $s_i = H(s_{i-1}, (i - 1, t_{i-1}))$. If so, it outputs $V(x, r, s_0, t_0, \dots, s_c, t_c)$.

Theorem 10.1 (Multi-input measure-and-reprogram [DFM20]). *Let c be an integer, and W, X, Y be finite sets. There exists a polynomial-time quantum algorithm S such that the following holds. Let A be an arbitrary quantum oracle algorithm that makes q queries to a uniformly random $H : (W \cup Y) \times X \to Y$ and outputs a tuple $(x_0, \dots, x_c)$. Then for any $\widehat{x} \in X^{c+1}$ without duplicate entries, any predicate V, and any $w \in W$,*

$$\Pr_{y_1, \dots, y_c}\left[(x_0, \dots, x_c) = \widehat{x} \wedge V(w, x_0, y_1, x_1, \dots, y_c, x_c) = 1 : (x_0, \dots, x_c) \leftarrow S^A(y_1, \dots, y_c)\right]$$

$$\geq \frac{c!}{(q + c + 1)^{2c}} \Pr_H\left[(x_0, \dots, x_c) = \widehat{x} \wedge V\begin{pmatrix} w, x_0, \\ y_1 := H(w, x_0), x_1, \\ y_2 := H(y_1, x_1), x_2, \dots, \\ y_c := H(y_{c-1}, x_{c-1}), x_c \end{pmatrix} = 1 \ : (x_0, \dots, x_c) \leftarrow A^H\right] - \epsilon_{\widehat{x}},$$

where $\sum_{\widehat{x}} \epsilon_{\widehat{x}} = c!/|Y|$, and S^A is an algorithm that has black-box access to the algorithms of A and for each $i \in [c]$, receives y_i and only after outputting x_{i-1}.[27]

Theorem 10.2. *If (P, V) is a sound protocol for $\mathcal{L}$, then $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ is a sound protocol for $\mathcal{L}$ in the quantum random oracle model.*

The following corollary then follows immediately from the protocol given in Section 9.3.

Corollary 10.3. *Assuming post-quantum indistinguishability obfuscation, the post-quantum hardness of the learning with errors problem, and post-quantum fully homomorphic encryption, there exists a designated verifier succinct non-interactive argument system (dvSNARG) for QMA in the quantum random oracle model.*

Proof. (of Theorem 10.2) Let $H : ((I \times S) \cup S) \times ([c] \times T) \to S$ be the random oracle used in $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$. Consider an adversary A in the protocol $(P_{\mathsf{FS}}, V_{\mathsf{FS}})$ that makes $q = \mathsf{poly}(\lambda)$ queries to H and consider any $x \notin \mathcal{L}$. For any r, let $V_{1,r}$ be the predicate V_1 with r hard-coded, and let A_r be the adversary A initialized with $(x, V_0(x, r))$. Define $\epsilon(r)$ to be the success probability of A_r (that is, the probability it makes V_{FS} output 1). Then for any fixed r,

$$\epsilon(r) = \Pr_H \left[V_{1,r} \begin{pmatrix} x, t_0, \\ s_1 := H((x, V_0(x,r), (0, t_0)), t_1, \ldots, \\ s_c := H(s_{c-1}, (c-1, t_{c-1})), t_c \end{pmatrix} = 1 : (t_0, \ldots, t_n) \leftarrow A_r^H \right].$$

Note that the overall success probability A is $\epsilon := \mathbb{E}_r[\epsilon(r)]$. Now, by setting $W = (I \times S), X = ([c] \times T), Y = S$, and $w = (x, V_0(x, r))$, Theorem 10.1 implies that for any fixed r, the success probability $\delta(r)$ of the simulator $S^{A_r}(s_1, \ldots, s_c)$ is

$$\delta(r) \geq \frac{1}{\mathsf{poly}(\lambda)} \epsilon(r) - \mathsf{negl}(\lambda),$$

which follows by (i) summing over all $t_0, \ldots, t_1$ and noting that $(0, t_0), \ldots, (c, t_c)$ contain no duplicates, and (ii) the fact that $q = \mathsf{poly}(\lambda)$ and c is a constant. Finally, observe that by the soundness of (P, V), $\mathbb{E}_r[\delta(r)] = \mathsf{negl}(\lambda)$. Indeed, by definition S^A is a valid cheating prover in the protocol (P, V) since it only receives random s_i after outputting t_{i-1}. This establishes that

$$\frac{1}{\mathsf{poly}(\lambda)} \mathop{\mathbb{E}}_r[\epsilon(r)] - \mathsf{negl}(\lambda) \leq \mathsf{negl}(\lambda),$$

which implies that $\epsilon = \mathsf{negl}(\lambda)$.

□

[27] This theorem as stated is actually a special case of [DFM20, Theorem 7], where w is fixed. In other words, it corresponds to [DFM20, Theorem 7] where the class of adversaries considered all produce a fixed w as the first part of their output.

10.2 Batch Arguments for QMA

Now, we show how to obtain batch arguments for QMA from the post-quantum hardness of learning with errors. We first describe a verifier-succinct protocol for verifying n QMA instances, where the verifier message size only grows with the time T needed for QMA verification of a *single* instance. Note that here we do *not* use the succinct key generation protocol from Section 7.1, and thus do not rely on indistinguishability obfuscation.

Ingredients:

- Let $(P_{\mathsf{FHM}}, V_{\mathsf{FHM}})$ be the non-interactive protocol described in Lemma 8.4 for language $\mathcal{L}$ with associated polynomials $k(\lambda), \ell(\lambda)$.
- Let $P_{\mathsf{Meas}} = (\mathsf{Commit}, \mathsf{Open})$ and $V_{\mathsf{Meas}} = (\mathsf{Gen}, \mathsf{Test}, \mathsf{Out})$ be the prover and verifier algorithms for an $\ell(\lambda)$-qubit commit-and-open measurement protocol, defined in Section 4 and constructed in [Mah18].

The Protocol:

- The verifier is initialized with n instances $(x_1, \ldots, x_n) \in \{0,1\}^\lambda$ and the prover is initialized with $(x_1, \ldots, x_n)$ and $k(\lambda)$ copies of each witness $|\phi_j\rangle \in \mathcal{R}_{\mathcal{L}}(x_j)$.
- The verifier samples $h \leftarrow \{0,1\}^{\ell(\lambda)}$ and defines C such that $C(i) = h_i$. The verifier computes $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, C)$ and sends pk to the prover.
- For each $j \in [n]$, the prover first computes $|\psi_j\rangle \leftarrow P_{\mathsf{FHM}}\left(x_j, |\phi_j\rangle^{\otimes k(\lambda)}\right)$, and then computes $(y_j, |\mathsf{st}_j\rangle) \leftarrow \mathsf{Commit}(\mathsf{pk}, |\psi_j\rangle)$. It sends $(y_1, \ldots, y_n)$ to the verifier.
- The verifier samples a random challenge $c \leftarrow \{0,1\}$ and sends c to the prover.
- For each $j \in [n]$, the prover computes $z_j \leftarrow \mathsf{Open}(|\mathsf{st}_j\rangle, c)$ and sends $(z_1, \ldots, z_n)$ to the verifier.
- If $c = 0$, the verifier checks whether $\mathsf{Test}(\mathsf{pk}, (y_j, z_j)) = \mathsf{acc}$ and rejects if the test fails on any index. If $c = 1$, the verifier computes $m_j \leftarrow \mathsf{Out}(\mathsf{sk}, (y_j, z_j))$ and checks whether $V_{\mathsf{FHM}}(x_j, m_j) = \mathsf{acc}$. The verifier accepts if and only if all of the verifications are successful.

Theorem 10.4. *Let $(P_{\mathsf{Batch}}, V_{\mathsf{Batch}})$ be the λ-fold parallel repetition of the above protocol. Then, $(P_{\mathsf{Batch}}, V_{\mathsf{Batch}})$ satisfies completeness as in Definition 3.1 and soundness as in Definition 3.2. Moreover, for instances $(x_1, \ldots, x_n)$ where T is the maximum QMA verification time for any individual x_i, the total size of verifier messages is $\mathsf{poly}(\lambda, T)$.*

Proof. First, the verifier message guarantee follows immediately from the description of the protocol. Completeness follows via the same argument used to prove completeness in Theorem 8.5 (without the additional step involving the PRF). Soundness also follows along the same lines, except that, if $x_i \notin \mathcal{L}$, we define $V^{(i)}_{\lambda,r,0}$ to be the verifier's accept projection *on instance* i on a test round, and $V^{(i)}_{\lambda,r,1}$ to be the verifier's accept projection *on instance* i on a measurement round. □

Finally, we observe that Theorem 9.3 holds when the verifier-succinct protocol is replaced with the batch protocol above, with no change in analysis. This results in the following corollary.

Corollary 10.5. *Assuming the post-quantum hardness of the learning with errors problem, there exists a batch argument for QMA, where the total communication is polynomial in the QMA verification time for a single QMA instance.*

10.3 Zero Knowledge

In this section, we provide sketches for how to obtain the following results.

- A (non-adaptive) *zero-knowledge* succinct argument for QMA (in the plain model).
- A (non-adaptive) *zero-knowledge* dvSNARG for QMA in the quantum random oracle model.

In both of our sketches, we will make use of secure two-party computation for *reactive* functionalities, which are interactive functionalities where multiple public circuits may be computed sequentially over private inputs, and where the description of these public circuits may be determined after some of the private inputs are submitted to the functionality and may even depend on the outputs of previously computed circuits.

The plain model. Here, we can start with the protocol in Section 9.2. This protocol as such does not provide any hiding property for the prover's witness. However, we can use secure two-party computation for the following reactive functionality to hide all information about the prover's witness from the verifier, while preserving soundness.

- Take as input random coins r_P, r_V from each party, compute the verifier's first message of the protocol using randomness $r := r_P \oplus r_V$, and output this message to the prover.
- For each subsequent round, take as input the prover's message, and then compute and output the next verifier's message (using random coins r) to the prover.
- After the final prover's message, compute and output the verifier's verdict (based on all prover messages and r) to the verifier.

Note that, since the verifier is classical, this functionality can be implemented by a protocol for (post-quantum) secure two-party computation of classical (reactive) functionalities, such as [HSS11].[28] To argue soundness (for any fixed no instance), we can run the two-party computation simulator for a malicious prover in order to extract inputs from the prover and reduce to soundness of the underlying protocol. To argue zero-knowledge (for any fixed yes instance), we can run the two-party computation simulator for a malicious verifier, programming the final output to 1. Note that the verifier only receives this single bit of information from the functionality, which is internally running an honest verifier. Thus, this simulation and output is indistinguishable from the real interaction with an honest prover.

[28] Note that [HSS11] is based on Watrous rewinding, and thus requires polynomially many rounds of interaction. We do not attempt to optimize the round-complexity of our zero-knowledge protocol, but note that non-black-box techniques such as those of [BS20] (with additional assumptions), or a relaxation to ϵ-zero-knowledge [CCLY21] could result in a constant-round protocol.

The QROM. In order to add zero-knowledge to our two-message succinct argument in the QROM, we have to be careful in order to avoid using the random oracle in a non-black-box manner. We achieve this in two steps: We first construct a constant-round honest-verifier zero-knowledge argument where, (i) the verifier is public-coin except for the first message, and (ii) the protocol remains zero-knowledge even against a verifier that computes its first message maliciously. Second, we compress this protocol into a two-message protocol in the QROM using the same arguments in Section 10.1. Since the protocol has constant rounds and is public-coin after the first verifier message, soundness holds via the same argument. Zero-knowledge holds because we have zero-knowledge against a malicious first message, and honest-verifier zero-knowledge with respect to all subsequent messages, which will be sampled uniformly at random by the random oracle.

We now turn our attention to the construction of the constant-round honest-verifier zero-knowledge argument. We are going to assume the existence of a post-quantum secure two-message two-party computation protocol for reactive (classical) functionalities. One can instantiate this with the two-message secure computation protocol of [IPS08] based on (post-quantum) two-message oblivious transfer in the common random string (CRS) model, which we can instantiate from the post-quantum hardness of learning with errors [PVW08]. Note that when we later compress this protocol in the QROM, the CRS can be sampled by querying the random oracle on a fixed input. We will use such a protocol to implement the following reactive functionality.

- Take as input random coins r from the verifier.
- The verifier's first message is sampled by the verifier given to the functionality as a *public input*.
- Take as input the prover's first message.
- The verifier's second message is sampled by the verifier and given to the functionality as a *public input*.
- ...
- Take as input the prover's final message.
- Check that the verifier's first message is computed honestly from random coins r, and if so, compute the verifier's verdict using r, the prover messages, and the verifier's messages, and deliver this output to the verifier.

Note that all the verifier messages are still sampled publicly and as in the protocol from Section 9.3, so the prover can still compute its responses given these messages.

Now, we argue that the resulting protocol satisfies the required properties. To argue soundness, we can run the two-party computation simulator for a malicious prover in order to extract inputs from the prover and reduce to soundness of the original protocol. To argue honest-verifier zero-knowledge with a malicious first message, we can run the two-party computation simulator for a malicious verifier, programming the final output to 1. However, since we are allowing the verifier to choose its first message maliciously, we have to argue that for *any* choice of randomness used to generate the verifier's first message, the subsequent interaction between honest prover (on input a valid witness for a true statement) and an honest verifier results in the verifier outputting 1 with overwhelming probability. Recalling the structure of the verifier's first message in the Section 9.3 protocol, we see that this requires perfectly correct FHE and a perfectly correct measurement protocol. Achieving FHE with perfect correctness is standard by truncating the error distribution, and we can obtain a perfectly correct measurement protocol as discussed in Section 3.5 and Section 4.1.

References

[ACGH20] Gorjan Alagic, Andrew M. Childs, Alex B. Grilo, and Shih-Han Hung. Non-interactive classical verification of quantum computation. pages 153–180, 2020. 3, 6, 13, 47, 48, 65

[ALM+92] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and hardness of approximation problems. pages 14–23, 1992. 1

[AS92] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs; A new characterization of NP. pages 2–13, 1992. 1

[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. pages 320–331, 2018. 18, 19, 20, 27, 28

[BDGM20] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Factoring and pairings are not necessary for io: Circular-secure lwe suffices. Cryptology ePrint Archive, Report 2020/1024, 2020. https://ia.cr/2020/1024. 22

[BFL90] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. pages 16–25, 1990. 1

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. pages 501–519, 2014. 22

[BGKW88] Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson. Multi-prover interactive proofs: How to remove intractability assumptions. pages 113–131, 1988. 1

[BGMZ18] James Bartusek, Jiaxin Guan, Fermi Ma, and Mark Zhandry. Return of GGH15: Provable security against zeroizing attacks. pages 544–574, 2018. 22

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. pages 309–325, 2012. 21

[BS20] Nir Bitansky and Omri Shmueli. Post-quantum zero knowledge in constant rounds. pages 269–279, 2020. 59

[BV97] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on computing*, 26(5):1411–1473, 1997. 1

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. pages 97–106, 2011. 21

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. pages 1–12, 2014. 21

[BV17] Nir Bitansky and Vinod Vaikuntanathan. A note on perfect correctness by derandomization. pages 592–606, 2017. 22

[BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. pages 280–300, 2013. 22

[CCLY21] Nai-Hui Chia, Kai-Min Chung, Xiao Liang, and Takashi Yamakawa. Post-quantum simulatable extraction with minimal assumptions: Black-box and constant-round. Cryptology ePrint Archive, Report 2021/1516, 2021. https://ia.cr/2021/1516. 59

[CCY20] Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. Classical verification of quantum computations with efficient verifier. pages 181–206, 2020. 3

[CLLW20] Kai-Min Chung, Yi Lee, Han-Hsuan Lin, and Xiaodi Wu. Constant-round blind classical verification of quantum sampling. *arXiv preprint arXiv:2012.04848*, 2020. 1

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments: breaking the quantum rewinding barrier. FOCS '21, 2021. 13, 14, 50, 51

[CVW18] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: Proofs, attacks, and candidates. pages 577–607, 2018. 22

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. pages 602–631, 2020. 56, 57

[DQV$^+$21] Lalita Devadas, Willy Quach, Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Succinct lwe sampling, random polynomials, and obfuscation. Cryptology ePrint Archive, Report 2021/1226, 2021. https://ia.cr/2021/1226. 22

[FHM18] Joseph F. Fitzsimons, Michal Hajdusek, and Tomoyuki Morimae. Post hoc verification of quantum computation. *Phys. Rev. Lett.*, 120:040501, Jan 2018. 2, 6, 13, 47, 48

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. pages 169–178, 2009. 21

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). pages 464–479, 1984. 22

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). pages 291–304, 1985. 1

[GP21] Romain Gay and Rafael Pass. Indistinguishability obfuscation from circular security. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 736–749. ACM, 2021. 22

[GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. pages 469–477, 2015. 4, 18

[HSS11] Sean Hallgren, Adam Smith, and Fang Song. Classical cryptographic protocols in a quantum world. pages 411–428, 2011. 59

[IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. pages 572–591, 2008. 60

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 60–73. ACM, 2021. 22

[Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). pages 723–732, 1992. 1

[KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. pages 669–684, 2013. 22

[LMS21] Alex Lombardi, Fermi Ma, and Nicholas Spooner. Post-quantum zero knowledge, revisited (or: How to do quantum rewinding undetectably). Cryptology ePrint Archive, Report 2021/1543, 2021. https://ia.cr/2021/1543. 14, 50, 51

[Mah18] Urmila Mahadev. Classical verification of quantum computations. pages 259–267, 2018. 1, 2, 3, 4, 5, 6, 7, 8, 16, 18, 19, 20, 23, 24, 25, 26, 27, 29, 41, 58

[Mic94] Silvio Micali. A secure and efficient digital signature algorithm. Technical Memo MIT/LCS/TM-501b, Massachusetts Institute of Technology, Laboratory for Computer Science, April 1994. 1

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. pages 554–571, 2008. 4, 18, 60

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. pages 187–196, 2008. 4, 18, 20

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. pages 84–93, 2005. 18

[SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. pages 475–484, 2014. 7, 22

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 135–152. Springer, 2012. 16

[Unr16a] Dominique Unruh. Collapse-binding quantum commitments without random oracles. pages 166–195, 2016. 20

[Unr16b] Dominique Unruh. Computationally binding quantum commitments. pages 497–527, 2016. 5, 20

[Vid20] Thomas Vidick. Interactions with quantum devices (course), 2020. http://users.cms.caltech.edu/~vidick/teaching/fsmp/fsmp.pdf. 2, 8, 9, 29, 35

[VZ21] Thomas Vidick and Tina Zhang. Classical proofs of quantum knowledge. pages 630–660, 2021. 6

[WW21] Hoeteck Wee and Daniel Wichs. Candidate obfuscation via oblivious LWE sampling. pages 127–156, 2021. 22

A Proofs from Section 8

Lemma A.1. *Consider a commit-challenge-response protocol with the following properties.*

1. *$V_{\lambda,r,0}$ does not depend on r (that is, it is publicly computable given the transcript).*
2. *For any P^*, if $\mathbb{E}_r\left[\left\langle\psi^{P^*}_{\lambda,r}\middle|\Pi^{P^*}_{\lambda,r,0}\middle|\psi^{P^*}_{\lambda,r}\right\rangle\right] = 1-\mathsf{negl}(\lambda)$, then $\mathbb{E}_r\left[\left\langle\psi^{P^*}_{\lambda,r}\middle|\Pi^{P^*}_{\lambda,r,1}\middle|\psi^{P^*}_{\lambda,r}\right\rangle\right] = \mathsf{negl}(\lambda)$.*

Then, the protocol has computationally orthogonal projectors.

Proof. Suppose there exists a prover P^* and a polynomial $p(\lambda)$ such that for infinitely many λ,

$$\mathop{\mathbb{E}}_r\left[\left\langle\psi^{P^*}_{\lambda,r}\middle|\Pi^{P^*}_{\lambda,r,0}\Pi^{P^*}_{\lambda,r,1}\Pi^{P^*}_{\lambda,r,0}\middle|\psi^{P^*}_{\lambda,r}\right\rangle\right] \geq 1/p(\lambda).$$

Define an alternate prover $\widehat{P}^*$ as follows.

1. $\widehat{P}^*$ takes as input $p(\lambda)^4$ copies of P^*'s auxiliary advice, and pk sampled by the verifier.
2. Repeat the following at most $p(\lambda)^4$ times:
   (a) Prepare the state $\left|\psi^{P^*}_{\lambda,r}\right\rangle$ using a copy of P^*'s auxiliary advice.
   (b) Apply the projective measurement $\left\{\Pi^{P^*}_{\lambda,r,0}, \mathbb{I}-\Pi^{P^*}_{\lambda,r,0}\right\}$, which is efficient due to property 1 of the commit-challenge-response protocol.
   (c) If the first outcome is observed, output the resulting state. Otherwise, repeat.
3. If $\widehat{P}^*$ has not terminated, output a dummy state $|\phi\rangle$ such that $\langle\phi|\,\Pi^{P^*}_{\lambda,r,0}\,|\phi\rangle = 1$.

Let $\left|\psi_{\lambda,r}^{\widehat{P}^*}\right\rangle$ be the state that results from the above procedure. Finally, let $\widehat{P}^*$ act identically to P^* after this point.

Next, let $\mathcal{R}_{\mathsf{term},\lambda} \coloneqq \left\{r : \left\langle \psi_{\lambda,r}^{P^*}\right| \Pi_{\lambda,r,0}^{P^*} \left|\psi_{\lambda,r}^{P^*}\right\rangle > 1/p(\lambda)^2\right\}$, and note that for any $r \in \mathcal{R}_{\mathsf{term},\lambda}$,

$$\Pr\left[\left|\psi_{\lambda,r}^{\widehat{P}^*}\right\rangle \neq \frac{\Pi_{\lambda,r,0}^{P^*}\left|\psi_{\lambda,r}^{P^*}\right\rangle}{\|\Pi_{\lambda,r,0}^{P^*}\left|\psi_{\lambda,r}^{P^*}\right\rangle\|}\right] \leq \left(1 - 1/p(\lambda)^2\right)^{p(\lambda)^4} \leq e^{-p(\lambda)^2} = \mathsf{negl}(\lambda).$$

Now, on the one hand,

$$\begin{aligned}
&\frac{1}{|\mathcal{R}|}\sum_{r \in \mathcal{R}_{\mathsf{term},\lambda}} \left\langle \psi_{\lambda,r}^{P^*}\right| \Pi_{\lambda,r,0}^{P^*}\Pi_{\lambda,r,1}^{P^*}\Pi_{\lambda,r,0}^{P^*}\left|\psi_{\lambda,r}^{P^*}\right\rangle \\
\leq &\frac{1}{|\mathcal{R}|}\sum_{r \in \mathcal{R}_{\mathsf{term},\lambda}} \left\langle \psi_{\lambda,r}^{\widehat{P}^*}\right| \Pi_{\lambda,r,1}^{\widehat{P}^*}\left|\psi_{\lambda,r}^{\widehat{P}^*}\right\rangle + \mathsf{negl}(\lambda) \\
\leq &\mathop{\mathbb{E}}_{r}\left[\left\langle \psi_{\lambda,r}^{\widehat{P}^*}\right| \Pi_{\lambda,r,1}^{\widehat{P}^*}\left|\psi_{\lambda,r}^{\widehat{P}^*}\right\rangle\right] + \mathsf{negl}(\lambda) \\
\leq &\mathsf{negl}(\lambda),
\end{aligned}$$

where the third inequality follows from property 2 of the commit-challenge-response protocol, since by definition $\mathbb{E}_r\left[\left\langle \psi_{\lambda,r}^{\widehat{P}^*}\right| \Pi_{\lambda,r,0}^{\widehat{P}^*}\left|\psi_{\lambda,r}^{\widehat{P}^*}\right\rangle\right] = 1$. On the other hand,

$$\begin{aligned}
&\frac{1}{|\mathcal{R}|}\sum_{r \in \mathcal{R}_{\mathsf{term},\lambda}} \left\langle \psi_{\lambda,r}^{P^*}\right| \Pi_{\lambda,r,0}^{P^*}\Pi_{\lambda,r,1}^{P^*}\Pi_{\lambda,r,0}^{P^*}\left|\psi_{\lambda,r}^{P^*}\right\rangle \\
= &\mathop{\mathbb{E}}_{r}\left[\left\langle \psi_{\lambda,r}^{P^*}\right| \Pi_{\lambda,r,0}^{P^*}\Pi_{\lambda,r,1}^{P^*}\Pi_{\lambda,r,0}^{P^*}\left|\psi_{\lambda,r}^{P^*}\right\rangle\right] - \frac{1}{|\mathcal{R}|}\sum_{r \notin \mathcal{R}_{\mathsf{term},\lambda}} \left\langle \psi_{\lambda,r}^{P^*}\right| \Pi_{\lambda,r,0}^{P^*}\Pi_{\lambda,r,1}^{P^*}\Pi_{\lambda,r,0}^{P^*}\left|\psi_{\lambda,r}^{P^*}\right\rangle \\
\geq &1/p(\lambda) - 1/p(\lambda)^2,
\end{aligned}$$

which is a contradiction, completing the proof. □

Theorem A.2 ([ACGH20]). *Consider the λ-fold parallel repetition of any commit-challenge-response protocol with computationally orthogonal projectors. The probability that the verifier accepts all λ parallel repetitions of the protocol is* $\mathsf{negl}(\lambda)$.

Proof. Let $\mathcal{R}$ be the randomness space of the single repetition protocol, and $r = (r_1, \ldots, r_\lambda) \in \mathcal{R}^{\otimes\lambda}$ be verifier randomness for the λ-fold parallel repetition. Now, any non-uniform prover P^* can be described by states $\left\{\left|\psi_{\lambda,r}^{P^*}\right\rangle\right\}_{\lambda,r}$ and families of unitaries $\left\{U_{\lambda,c}^{P^*}\right\}_{\lambda,c}$, where $c \in \{0,1\}^\lambda$ ranges over all of the verifier challenges.

For each $c \in \{0,1\}^\lambda$, define

$$\Pi_{\lambda,r,c}^{P^*} \coloneqq U_{\lambda,c}^{P^*\dagger}\left(V_{\lambda,r_1,c_1} \otimes \cdots \otimes V_{\lambda,r_\lambda,c_\lambda}\right)U_{\lambda,c}^{P^*}.$$

Claim A.3. *For any $c_1 \neq c_2 \in \{0,1\}^\lambda$,*

$$\mathop{\mathbb{E}}_{r}\left[\left\langle \psi_{\lambda,r}^{P^*}\right| \Pi_{\lambda,r,c_2}^{P^*}\Pi_{\lambda,r,c_1}^{P^*} + \Pi_{\lambda,r,c_1}^{P^*}\Pi_{\lambda,r,c_2}^{P^*}\left|\psi_{\lambda,r}^{P^*}\right\rangle\right] = \mathsf{negl}(\lambda).$$

Proof. Suppose there exists $i \in [\lambda]$ such that $(c_1)_i = 1$ and $(c_2)_i = 0$ (the other case is symmetric). Since for any quantum state $|\psi\rangle$ and two projectors Π_1, Π_2,

$$\langle\psi|\, \Pi_2\Pi_1 + \Pi_1\Pi_2 \,|\psi\rangle \leq 2|\, \langle\psi|\, \Pi_2\Pi_1 \,|\psi\rangle \,| \leq 2\, \langle\psi|\, \Pi_2\Pi_1\Pi_2 \,|\psi\rangle^{1/2},$$

it then suffices (by Jensen's inequality) to show that

$$\mathop{\mathbb{E}}_{r}\left[\left\langle \psi^{P^*}_{\lambda,r} \middle| \Pi^{P^*}_{\lambda,r,c_2}\Pi^{P^*}_{\lambda,r,c_1}\Pi^{P^*}_{\lambda,r,c_2} \middle| \psi^{P^*}_{\lambda,r} \right\rangle\right] = \mathsf{negl}(\lambda).$$

To see this, let

$$V^{(i)}_{\lambda,r_i,b} \coloneqq \mathbb{I} \otimes \cdots \otimes \mathbb{I} \otimes V_{\lambda,r_i,b} \otimes \mathbb{I} \otimes \cdots \otimes \mathbb{I},$$

for $i \in [\lambda], b \in \{0,1\}$, and observe that

$$\begin{aligned}
&\mathop{\mathbb{E}}_{r}\left[\left\langle \psi^{P^*}_{\lambda,r} \middle| \Pi^{P^*}_{\lambda,r,c_2}\Pi^{P^*}_{\lambda,r,c_1}\Pi^{P^*}_{\lambda,r,c_2} \middle| \psi^{P^*}_{\lambda,r} \right\rangle\right] \\
\leq &\mathop{\mathbb{E}}_{r}\left[\left\langle \psi^{P^*}_{\lambda,r} \middle| {U^{P^*}_{\lambda,c_2}}^\dagger V^{(i)}_{\lambda,r_i,0} U^{P^*}_{\lambda,c_2} {U^{P^*}_{\lambda,c_1}}^\dagger V^{(i)}_{\lambda,r_i,1} U^{P^*}_{\lambda,c_1} {U^{P^*}_{\lambda,c_2}}^\dagger V^{(i)}_{\lambda,r_i,0} U^{P^*}_{\lambda,c_2} \middle| \psi^{P^*}_{\lambda,r} \right\rangle\right] \\
= &\mathop{\mathbb{E}}_{r_i}\left[\left\langle \widehat{\psi}^{P^*}_{\lambda,r_i} \middle| {U^{P^*}_{\lambda,c_2}}^\dagger V^{(i)}_{\lambda,r_i,0} U^{P^*}_{\lambda,c_2} {U^{P^*}_{\lambda,c_1}}^\dagger V^{(i)}_{\lambda,r_i,1} U^{P^*}_{\lambda,c_1} {U^{P^*}_{\lambda,c_2}}^\dagger V^{(i)}_{\lambda,r_i,0} U^{P^*}_{\lambda,c_2} \middle| \widehat{\psi}^{P^*}_{\lambda,r_i} \right\rangle\right] \\
= &\mathsf{negl}(\lambda),
\end{aligned}$$

where for each $r_i \in \mathcal{R}$, $\left|\widehat{\psi}^{P^*}_{\lambda,r_i}\right\rangle$ is the purification of the mixed state (written in ensemble form)

$$\left\{\frac{1}{|\mathcal{R}|^{\lambda-1}}, \left|\psi^{P^*}_{\lambda,(r_1,\ldots,r_\lambda)}\right\rangle\right\}_{(r_1,\ldots,r_{i-1},r_{i+1},\ldots,r_\lambda)\in\mathcal{R}^{\otimes\lambda-1}},$$

and the final equality follows from the computational orthogonal projectors property of the commit-challenge-response protocol. Indeed, one can define an efficient prover P_i^* for the i'th iteration of the commit-challenge-response protocol by defining $U^{P_i^*}_{\lambda,0} \coloneqq U^{P^*}_{\lambda,c_2}$ and $U^{P_i^*}_{\lambda,1} \coloneqq U^{P^*}_{\lambda,c_1}$ and noting that $\left|\widehat{\psi}^{P^*}_{\lambda,r_i}\right\rangle$ is efficient to prepare while interacting with the i'th iteration of $\mathcal{V}$, by running P^* and $\lambda - 1$ coherently executed copies of $\mathcal{V}$.

□

Now observe that the probability the verifier accepts the parallel repeated protocol is

$$\begin{aligned}
&\frac{1}{2^\lambda}\mathop{\mathbb{E}}_{r}\left[\left\langle\psi^{P^*}_{\lambda,r}\right|\sum_{c\in\{0,1\}^\lambda}\Pi^{P^*}_{\lambda,c,r}\left|\psi^{P^*}_{\lambda,r}\right\rangle\right]\\
\leq&\frac{1}{2^\lambda}\mathop{\mathbb{E}}_{r}\left[\left(\left\langle\psi^{P^*}_{\lambda,r}\right|\left(\sum_{c\in\{0,1\}^\lambda}\Pi^{P^*}_{\lambda,c,r}\right)^2\left|\psi^{P^*}_{\lambda,r}\right\rangle\right)^{1/2}\right]\\
\leq&\frac{1}{2^\lambda}\mathop{\mathbb{E}}_{r}\left[\left(\sum_{c\in\{0,1\}^\lambda}\left\langle\psi^{P^*}_{\lambda,r}\right|\Pi^{P^*}_{\lambda,c,r}\left|\psi^{P^*}_{\lambda,r}\right\rangle\right)^{1/2}\right]\\
&+\frac{1}{2^\lambda}\left(\sum_{\{c_1,c_2\}\in(\{0,1\}^\lambda)^2}\mathop{\mathbb{E}}_{r}\left[\left\langle\psi^{P^*}_{\lambda,r}\right|\Pi^{P^*}_{\lambda,c_2,r}\Pi^{P^*}_{\lambda,c_1,r}+\Pi^{P^*}_{\lambda,c_1,r}\Pi^{P^*}_{\lambda,c_2,r}\left|\psi^{P^*}_{\lambda,r}\right\rangle\right]\right)^{1/2}\\
\leq&\frac{1}{2^{\lambda/2}}+\frac{1}{2^\lambda}\left(\sum_{\{c_1,c_2\}\in(\{0,1\}^\lambda)^2}\mathsf{negl}(\lambda)\right)^{1/2}=\mathsf{negl}(\lambda),
\end{aligned}$$

where the first inequality holds because $\left|\psi^{P^*}_{\lambda,r}\right\rangle\left\langle\psi^{P^*}_{\lambda,r}\right|\preceq\mathbb{I}$, the second inequality uses Jensen's inequality and the fact that projectors are idempotent, and the third inequality follows from Claim A.3.

□

B Proof of Claim 6.4

We now prove Claim 6.4, which is restated below for convenience.

Claim B.1. *For all $(u,v)\in\{0,1\}^R\times\{0,1\}^S$ it holds that*

$$\mathrm{Tr}\left(\Pi^{\sigma_x}_u\Pi^{\sigma_z}_v\boldsymbol{\tau}\right)=\mathop{\mathbb{E}}_{u'\in\{0,1\}^R}\langle\psi|\,\Pi^Z_vZ(u')\Pi^X_{u'\oplus u}Z(u')\Pi^Z_v\,|\psi\rangle\,. \tag{7}$$

Proof. Using the definition of $\boldsymbol{\tau}$, we get

$$\begin{aligned}
\mathrm{Tr}(\Pi^{\sigma_x}_u\Pi^{\sigma_z}_v\boldsymbol{\tau})=2^{-2N}&\sum_{r',s',r'',s''\in\{0,1\}^N}\Big(\langle\psi|\,Z(s')X(r'\oplus r'')Z(s'')\,|\psi\rangle_{\mathcal{H}}\\
&\left\langle\phi^+\right|^{\otimes N}\left(\sigma_z(s')\sigma_x(r'\oplus r'')\sigma_z(s'')\right)_{\mathcal{A}_1}\otimes\left(\Pi^{\sigma_x}_u\Pi^{\sigma_z}_v\right)_{\mathcal{A}_2}\left|\phi^+\right\rangle^{\otimes N}\Big)\\
=2^{-2N}&\sum_{r',s',r'',s''\in\{0,1\}^N}(-1)^{(r'\oplus r'')\cdot s''}\Big(\langle\psi|\,Z(s')X(r'\oplus r'')Z(s'')\,|\psi\rangle_{\mathcal{H}}\\
&\left\langle\phi^+\right|^{\otimes N}\left(\sigma_z(s'\oplus s'')\sigma_x(r'\oplus r'')\right)_{\mathcal{A}_1}\otimes\left(\Pi^{\sigma_x}_u\Pi^{\sigma_z}_v\right)_{\mathcal{A}_2}\left|\phi^+\right\rangle^{\otimes N}\Big).
\end{aligned}\tag{8}$$

However, most of the terms in Eq. (8) are zero: observe that when $(r'\oplus r'')_j\neq0$ for any $j\in S$, or $(s'\oplus s'')_j\neq0$ for any $j\in R$, we have

$$\left\langle\phi^+\right|^{\otimes N}\left(\sigma_z(s'\oplus s'')\sigma_x(r'\oplus r'')\right)_{\mathcal{A}_1}\otimes\left(\sigma_x(u)\sigma_z(v)\right)_{\mathcal{A}_2}\left|\phi^+\right\rangle^{\otimes N}=0.$$

We can therefore rewrite Eq. (8) using the following change of variables:

- Since $s' \oplus s''$ must be 0 on R, the restriction of s' and s'' to R must be the same vector $u' \in \{0,1\}^R$. Let the restriction of s' and s'' to indices in S be $v', v'' \in \{0,1\}^S$ respectively.
- Since $r' \oplus r''$ must be 0 on S, let $u'' \in \{0,1\}^R$ denote the restriction of $r' \oplus r''$ to indices in R. Note that for each u'', there are 2^N choices of (r', r'') satisfying $u'' = r' \oplus r''$.

By a straightforward calculation, we have for all $u'' \in \{0,1\}^R$ and all $s', s'' \in \{0,1\}^N$ that

$$\sum_{\substack{r',r'' \in \{0,1\}^N \\ (r' \oplus r'') = u''}} \langle \phi^+ |^{\otimes N} \left(\sigma_z(s' \oplus s'') \sigma_x(r' \oplus r'') \right)_{\mathcal{A}_1} \otimes \left(\Pi_u^{\sigma_x} \Pi_v^{\sigma_z} \right)_{\mathcal{A}_2} | \phi^+ \rangle^{\otimes N} = (-1)^{u'' \cdot u + (s' \oplus s'')v} .$$

Plugging this into Eq. (8), and using the fact that $(-1)^{(s' \oplus s'')v} = (-1)^{(v' \oplus v'')v}$, we obtain

$$\begin{aligned} \mathrm{Tr}(\Pi_u^{\sigma_x} \Pi_v^{\sigma_z} \boldsymbol{\tau}) &= 2^{-2N} \sum_{\substack{u',u'' \in \{0,1\}^R \\ v',v'' \in \{0,1\}^S}} (-1)^{(u \oplus u') \cdot u'' + (v' \oplus v'')v} \Big(\langle \psi | \, Z(v') Z(u') X(u'') Z(u') Z(v'') \, | \psi \rangle_{\mathcal{H}} \Big) \\ &= \mathop{\mathbb{E}}_{u' \in \{0,1\}^R} \langle \psi | \, \Pi_v^Z Z(u') \Pi_{u \oplus u'}^X Z(u') \Pi_v^Z \, | \psi \rangle \end{aligned}$$

where the second equality follows from plugging in the definitions of Π_v^Z and $\Pi_{u \oplus u'}^X$. □

C Proof of Claim 6.7

We now prove Claim 6.7, which we restate below for convenience.

Claim C.1. *Let $k = k(\lambda)$ be a positive integer-valued function of a security parameter λ. Let $\{D_{0,\lambda}\}_{\lambda \geq 1}$ and $\{D_{1,\lambda}\}_{\lambda \geq 1}$ be families of distributions on $\{0,1\}^{k+1}$ such that the marginal distributions $D'_{0,\lambda}$ and $D'_{1,\lambda}$ of $D_{0,\lambda}$ and $D_{1,\lambda}$ respectively on the first k bits are computationally indistinguishable. Suppose that $D_{0,\lambda}$ and $D_{1,\lambda}$ are computationally* distinguishable. *Then there is an efficiently computable binary-outcome POVM $\{M, \mathsf{Id} - M\}$ acting on k qubits such that*

$$\left| \mathop{\mathbb{E}}_{x \sim D_{0,\lambda}} (-1)^{x_{k+1}} \langle x_{\leq k} | \, M \, | x_{\leq k} \rangle - \mathop{\mathbb{E}}_{x \sim D_{1,\lambda}} (-1)^{x_{k+1}} \langle x_{\leq k} | \, M \, | x_{\leq k} \rangle \right| > \frac{1}{\mathsf{poly}(\lambda)} .$$

Proof. By assumption there exists an efficient distinguisher between D_0 and D_1 (for simplicity we omit the index λ from the notation). Let A be a circuit for the distinguisher: A has $(k+1)$ input qubits as well as m ancilla qubits, and a designated output qubit. Let Π_1 be the projection on the output qubit being equal to 1. Suppose without loss of generality that

$$\mathop{\mathbb{E}}_{x \sim D_0} \langle x, 0^m | \, A^\dagger \Pi_1 A \, | x, 0^m \rangle > \mathop{\mathbb{E}}_{x \sim D_1} \langle x, 0^m | \, A^\dagger \Pi_1 A \, | x, 0^m \rangle + \frac{1}{q}, \tag{9}$$

for some polynomial $q = q(\lambda)$. Letting $|b\rangle_{k+1}$ denote the $(k+1)$-st qubit, we can write

$$\begin{aligned} \mathop{\mathbb{E}}_{x \sim D_1} \langle x, 0^m | \, A^\dagger \Pi_1 A \, | x, 0^m \rangle = & \mathop{\mathbb{E}}_{x \sim D_1} x_{k+1} \langle x_{\leq k}, 0^m | \, \langle 1 |_{k+1} \, A^\dagger \Pi_1 A \, | 1 \rangle_{k+1} \, | x_{\leq k}, 0^m \rangle \\ & + \mathop{\mathbb{E}}_{x \sim D_1} (1 - x_{k+1}) \langle x_{\leq k}, 0^m | \, \langle 0 |_{k+1} \, A^\dagger \Pi_1 A \, | 0 \rangle_{k+1} \, | x_{\leq k}, 0^m \rangle . \end{aligned}$$

Let $M_b := \langle b, 0^m | A^\dagger \Pi_1 A | b, 0^m \rangle$ (where b corresponds to the $(k+1)$-st qubit); note that M_b is a positive semi-define. We can rewrite the right-hand-side as

$$\mathop{\mathbb{E}}_{x \sim D_1} x_{k+1} \langle x_{\leq k} | (M_1 - M_0) | x_{\leq k} \rangle + \mathop{\mathbb{E}}_{x \sim D_1} \langle x_{\leq k} | M_0 | x_{\leq k} \rangle . \tag{10}$$

Using a similar expansion while taking the expectation under D_0 yields

$$\mathop{\mathbb{E}}_{x \sim D_0} x_{k+1} \langle x_{\leq k} | (M_1 - M_0) | x_{\leq k} \rangle + \mathop{\mathbb{E}}_{x \sim D_0} \langle x_{\leq k} | M_0 | x_{\leq k} \rangle . \tag{11}$$

Plugging Eqs. (10) and (11) into Eq. (9) gives

$$\begin{aligned} &\mathop{\mathbb{E}}_{x \sim D_0} x_{k+1} \langle x_{\leq k} | (M_1 - M_0) | x_{\leq k} \rangle - \mathop{\mathbb{E}}_{x \sim D_1} x_{k+1} \langle x_{\leq k} | (M_1 - M_0) | x_{\leq k} \rangle \\ &\qquad > \mathop{\mathbb{E}}_{x \sim D_1} \langle x_{\leq k} | M_0 | x_{\leq k} \rangle - \mathop{\mathbb{E}}_{x \sim D_0} \langle x_{\leq k} | M_0 | x_{\leq k} \rangle + \frac{1}{q}. \end{aligned}$$

For $b \in \{0, 1\}$, note that $\{M_b, \mathsf{Id} - M_b\}$ is an efficiently computable POVM since it can be performed by initializing the $(k+1)$-st qubit to $|b\rangle$, the ancilla qubits to $|0^m\rangle$, applying A, and measuring whether the output qubit is 1. Since D_0' and D_1' are computationally indistinguishable, we have

$$\begin{aligned} &\mathop{\mathbb{E}}_{x \sim D_0} x_{k+1} \langle x_{\leq k}, 0^m | (M_1 - M_0) | x_{\leq k}, 0^m \rangle - \mathop{\mathbb{E}}_{x \sim D_1} x_{k+1} \langle x_{\leq k}, 0^m | (M_1 - M_0) | x_{\leq k}, 0^m \rangle \\ &\qquad > \frac{1}{q} - \mathsf{negl}(\lambda). \end{aligned}$$

We observe that there must exist $b \in \{0, 1\}$ such that when $M = M_b$, we have

$$\begin{aligned} &\left| \mathop{\mathbb{E}}_{x \sim D_0} x_{k+1} \langle x_{\leq k}, 0^m | M | x_{\leq k}, 0^m \rangle - \mathop{\mathbb{E}}_{x \sim D_1} x_{k+1} \langle x_{\leq k}, 0^m | M | x_{\leq k}, 0^m \rangle \right| \\ &\qquad > \frac{1}{\mathsf{poly}(\lambda)}. \end{aligned}$$

Finally, by plugging in the identity $(-1)^b = 1 - 2b$ for $b \in \{0, 1\}$ and appealing once again to the indistinguishability of D_0' and D_1', we conclude that

$$\begin{aligned} &\left| \mathop{\mathbb{E}}_{x \sim D_0} (-1)^{x_{k+1}} \langle x_{\leq k}, 0^m | M | x_{\leq k}, 0^m \rangle - \mathop{\mathbb{E}}_{x \sim D_1} (-1)^{x_{k+1}} \langle x_{\leq k}, 0^m | M | x_{\leq k}, 0^m \rangle \right| \\ &\qquad > \frac{1}{\mathsf{poly}(\lambda)}. \end{aligned}$$

□