Classical zero-knowledge arguments for quantum computations

Thomas Vidick[*] Tina Zhang[†]

Abstract

We show that every language in BQP admits a classical-verifier, quantum-prover zero-knowledge argument system which is sound against quantum polynomial-time provers and zero-knowledge for classical (and quantum) polynomial-time verifiers. The protocol builds upon two recent results: a computational zero-knowledge proof system for languages in QMA, with a quantum verifier, introduced by Broadbent et al. (FOCS 2016), and an argument system for languages in BQP, with a classical verifier, introduced by Mahadev (FOCS 2018).

1 Introduction

The paradigm of the interactive proof system is a versatile tool in complexity theory. Although traditional complexity classes are usually defined in terms of a single Turing machine—NP, for example, can be defined as the class of languages which a non-deterministic Turing machine is able to decide—many have reformulations in the language of interactive proofs, and such reformulations often inspire natural and fruitful variants on the traditional classes upon which they are based. (The class MA, for example, can be considered a natural extension of NP under the interactive-proof paradigm.)

Intuitively speaking, an interactive proof system is a model of computation involving two entities, a *verifier* and a *prover*, the former of whom is computationally efficient, and the latter of whom is unbounded but untrusted. The verifier and the prover exchange messages, and the prover attempts to 'convince' the verifier that a certain problem instance is a yes-instance. We can define some particular complexity class as the set of languages for which there exists an interactive proof system that 1) is *complete*, 2) is *sound*, and 3) has certain other properties which vary depending on the class in question. Completeness means, in this case, that for any problem instance in the language, there is an interactive proof involving r messages in total that the prover can offer the verifier which will cause it to accept with at least some probability p; and soundness means that, for any problem instance not in the language, no prover can cause the verifier to accept, except with some small probability q. For instance, if we require that the verifier is a deterministic polynomial-time Turing machine, and set $r = 1, p = 1$, and $q = 0$, the class that we obtain is of course the class NP. If we allow the verifier to be a probabilistic polynomial-time machine, and set $r = 1, p = \frac{2}{3}$, $q = \frac{1}{3}$, we have MA. Furthermore, if we allow the verifier to be an efficient *quantum* machine, and we allow the prover to communicate with it quantumly, but we retain the parameter settings from MA, we obtain the class QMA. Finally, if we allow r to be any polynomial in n, where n is the size of the problem instance, but otherwise preserve the parameter settings from MA, we obtain the class IP.

For every complexity class thus defined, there are two natural subclasses which consist of the languages that admit, respectively, a *statistical* and a *computational zero-knowledge* interactive proof system with otherwise the same properties. The notion of a zero-knowledge proof system was first considered by Goldwasser, Micali

[*]Department of Computing and Mathematical Sciences, California Institute of Technology, USA. `vidick@cms.caltech.edu`
[†]Department of Physics, California Institute of Technology, USA. `tinazhang@caltech.edu`

and Rackoff in [GMR89], and formalises the surprising but powerful idea that the prover may be able to prove statements to the verifier in such a way that the verifier learns nothing except that the statements are true. Informally, an interactive proof system is *statistical zero-knowledge* if an arbitrary malicious verifier is able to learn from an honest prover that a problem instance is a yes-instance, but can extract only negligible amounts of information from it otherwise; and the computational variant provides the same guarantee only for malicious polynomial-time verifiers. For IP in particular, the subclass of languages which admit a statistical zero-knowledge proof system that otherwise shares the same properties had by proof systems for languages in IP is known by the abbreviation SZK. Its computational sibling, meanwhile, is known by the abbreviation CZK. It is well-known that, contingent upon the existence of one-way functions, NP $\subseteq$ CZK: computational zero-knowledge proof systems have been known to exist for every language in NP since the early 1990s ([GMW91]). However, because these proof systems often relied upon intractability assumptions or techniques (e.g. 'rewinding') that failed in quantum settings, it was not obvious until recently how to obtain an analogous result for QMA. One design for a zero-knowledge proof system for languages in QMA was introduced by Broadbent, Ji, Song and Watrous in [BJSW16]. Their work establishes that, provided that a quantum computationally concealing, unconditionally binding commitment scheme exists, QMA $\subseteq$ QCZK.

There are, of course, a myriad more variations on the theme of interactive proofs in the quantum setting, each of which defines another complexity class. For example, motivated partly by practical applications, one might also consider the class of languages which can be decided by an interactive proof system involving a classical verifier and a quantum prover communicating classically, in which the soundness condition still holds against arbitrary provers, but the honest prover can be implemented in quantum polynomial time. (For simplicity, we denote this class by IP_{BQP}.) The motivation for this specific set of criteria is as follows: large-scale quantum devices are no longer so distant a dream as they seemed only a decade ago. If and when we have such devices, how will we verify, using our current generation of classical devices, that our new quantum computers can indeed decide problems in BQP? This problem—namely, the problem of showing that BQP $\subseteq \text{IP}_{\text{BQP}}$—is known informally as the problem of *quantum verification.*

The problem of quantum verification has not yet seen a solution, but in recent years a number of strides have been made toward producing one. As of the time of writing, protocols are known for the following three variants on the problem:

1. It was shown in [ABE10, ABOEM17] and [BFK09, FK17] that a classical verifier holding a *single-qubit quantum register* can decide languages in BQP by communicating quantumly with a single BQP prover. The protocol presented for this purpose is sound against arbitrary provers.
2. It was shown in [RUV13] that an entirely classical verifier can decide languages in BQP by interacting classically with two entangled, non-communicating BQP provers. This protocol is likewise sound against arbitrary provers.
3. It was shown in [Mah18] that an entirely classical verifier can decide languages in BQP by executing an *argument system* with a single BQP prover. (An argument system differs from a proof system in that it need not be sound against arbitrary provers, but only computationally bounded ones. In this case, the argument system in [Mah18] is sound against quantum polynomial-time provers. The class of languages for which there exists an argument system involving a classical probabilistic polynomial-time verifier and a quantum polynomial-time prover is referred to throughout [Mah18] as QPIP_0.) The argument system introduced in [Mah18] is reliant upon cryptographic assumptions about the quantum intractability of Learning With Errors (LWE; see [Reg09]) for its soundness. For practical purposes, if this assumption holds true, the problem of verification can be considered solved.

The last of these three results establishes that BQP $\subseteq \text{QPIP}_0$, contingent upon the intractability of LWE. In this work, we show that the protocol which [Mah18] introduces for this purpose can be combined with the zero-knowledge proof system for QMA presented in [BJSW16] in order to obtain a *zero-knowledge* argument system for BQP. It follows naturally that, if the LWE assumption holds, and quantum computationally hiding, unconditionally binding commitment schemes exist, BQP $\subseteq$ CZK-QPIP$_0$, where the latter refers to the class of languages for which there exists a *computational zero-knowledge* interactive argument system involving a classical verifier and a quantum polynomial-time prover. Zero-knowledge protocols for languages in NP are an essential component of many cryptographic constructions, such as identification schemes, and are often used in general protocol design (for example, one can force a party to follow a prescribed protocol by requiring it to produce a zero-knowledge proof that it did so). Our result opens the door for the use of zero-knowledge proofs in protocols involving classical and quantum parties which interact classically in order to decide languages defined in terms of quantum information (for instance, to verify that one of the parties possesses a quantum state having certain properties).

We now briefly describe our approach to the problem. The proof system for languages in QMA presented in [BJSW16] is *almost* classical, in the sense that the only quantum action which the honest verifier performs is to measure a quantum state after applying Clifford gates to it. The key contribution which [Mah18] makes to the problem of verification is to introduce a *measurement protocol* which, intuitively, allows a classical verifier to obtain honest measurements of its prover's quantum state. The combining of the proof system from [BJSW16] and the measurement protocol from [Mah18] is therefore a fairly natural action.

That the proof system of [BJSW16] is complete for languages in QMA follows from the QMA-completeness of a problem which the authors term the *5-local Clifford Hamiltonian problem*. However, the argument system which [Mah18] presents relies upon the QMA-completeness of the well-known *2-local XZ Hamiltonian problem* (see Definition 2.1). For this reason, the two results cannot be composed directly. Our first step is to make some modifications to the protocol introduced in [BJSW16] so it can be used to verify that an XZ Hamiltonian is satisfied, instead of verifying that a Clifford Hamiltonian is satisfied. We then introduce a composite protocol which replaces the quantum measurement in the protocol from [BJSW16] with an execution of the measurement protocol from [Mah18]. With the eventual object in mind of proving that the result is sound and zero-knowledge, we introduce a *trapdoor check* step into our composite protocol, and split the *coin-flipping protocol* used in the proof system from [BJSW16] into two stages. We explain these decisions briefly here, after we present a summary of our protocol, and refer the reader to Sections 3 and 5 for fuller expositions.

Protocol 1.1. *Zero-knowledge, classical-verifier argument system for BQP (informal summary).*

Parties.

The protocol involves

1. A *verifier*, which runs in classical probabilistic polynomial time;
2. A *prover*, which runs in quantum polynomial time.

Inputs. The protocol requires the following primitives:

- A perfectly binding, quantum computationally concealing commitment protocol.
- A zero-knowledge proof system for NP.
- An extended trapdoor claw-free function family (ETCFF family), as defined in [Mah18].

Apart from the above cryptographic primitives, we assume that the verifier and the prover also receive the following inputs.

1. Input to the verifier: a 2-local XZ Hamiltonian H (see Definition 2.1), along with two numbers, a and b, which define a promise about the ground energy of H. Because the 2-local XZ Hamiltonian promise problem is complete for QMA, and BQP $\subseteq$ QMA, any input to any decision problem in BQP can be reduced to an instance of the 2-local XZ Hamiltonian problem.

2. Input to the prover: the Hamiltonian H, the numbers a and b, and the quantum state $\rho = \sigma^{\otimes m}$, where σ is a ground state of the Hamiltonian H.

Protocol.

1. The prover applies an encoding process to ρ. Informally, the encoding can be thought of as a combination of an encryption scheme and an authentication scheme: it both hides the witness state ρ and ensures that the verifier cannot meaningfully tamper with the measurement results that it reports in step 5. Like most encryption and authentication schemes, this encoding scheme is keyed. For convenience, we refer to the encoding procedure determined by a particular encoding key K as E_K.[1]

2. The prover commits to the encoding key K from the previous step using a classical commitment protocol, and sends the resulting commitment string z to the verifier.

3. The verifier and the prover jointly decide which random terms from the Hamiltonian H the verifier will check by executing a coin-flipping protocol. ('Checking terms of H' means that the verifier obtains measurements of the state $E_K(\rho)$ and checks that the outcomes are distributed a particular way—or, alternatively, asks the prover to prove to it that they are.) However, because it is important that the prover does not know which terms will be checked before the verifier can check them, the two parties only execute the first half of the coin-flipping protocol at this stage. The verifier commits to its part of the random string, r_v, and sends the resulting commitment string to the prover; the prover sends the verifier r_p, its own part of the random string; and the verifier keeps the result of the protocol $r = r_v \oplus r_p$ secret for the time being. The random terms in the Hamiltonian which the verifier will check are determined by r.

4. The verifier and the prover execute the measurement protocol from [Mah18]. Informally, this allows the verifier to obtain honest measurements of the qubits of the prover's encoded witness state, so that it can check the Hamiltonian term determined by r. The soundness guarantee of the measurement protocol prevents the prover from cheating, even though the prover, rather than the verifier, is physically performing the measurements. This soundness guarantee relies on the security properties of a family of trapdoor one-way functions termed an ETCFF family in [Mah18]. Throughout the measurement protocol, the verifier holds trapdoors for these one-way functions, but the prover does not, and this asymmetry is what allows the (intrinsically weaker) verifier to ensure that the prover does not cheat.

5. The verifier opens its commitment to r_v, and also sends the prover its measurement outcomes u and function trapdoors from the previous step.

6. The prover checks, firstly, that the verifier's trapdoors are valid, and that it did not tamper with the measurement outcomes u. (It can determine the latter by making use of the authentication-scheme-like properties of E_K from step 1.) If both tests pass, it then proves the following statement to the verifier, using a zero-knowledge proof system for NP:

   There exists a string s_p and an encoding key K such that $z = \mathsf{commit}(K, s_p)$ and $Q(K, r, u) = 1$.

   The function Q is a predicate which, intuitively, takes the value 1 if and only if both the verifier *and* the prover were honest. In more specific (but still informal) terms, $Q(K, r, u)$ takes the value 1 if u contains the outcomes of honest measurements of the state $E_K(\rho)$, where ρ is a state that passes the set of Hamiltonian energy tests determined by r.

The reason we delay the verifier's reveal of r_v (rather than completing the coin-flipping in one step, as is done in the protocol in [BJSW16]) is fairly easily explained. In our classical-verifier protocol, the prover cannot physically send the quantum state $E_K(\rho)$ to its verifier before the random string r is decided, as the prover of the protocol in [BJSW16] does. If we allow our prover to know r at the time when it performs measurements on the witness ρ, it will trivially be able to cheat.

[1]The notation used here for the encoding key is not consistent with that which is used later on; it is simplified for the purposes of exposition.

The trapdoor check, meanwhile, is an addition which we make because we wish to construct a *classical* simulator for our protocol when we prove that it is zero-knowledge. Since our verifier is classical, we need to achieve a classical simulation of the protocol in order to prove that its execution (in yes-instances) does not impart to the verifier any knowledge it could not have generated itself. During the measurement protocol, however, the prover is required to perform quantum actions which no classical polynomial-time algorithm could simulate unless it had access to the verifier's function trapdoors. Naturally, we cannot ask the verifier to reveal its trapdoors before the measurement protocol takes place. As such, we ask the verifier to reveal them immediately afterwards instead, and show in Section 5 that this (combined with the encryption-scheme properties of the prover's encoding E_K) allows us to construct a classical simulator for Protocol 1.1 in yes-instances.

The organisation of the paper is as follows.

1. Section 2 ('Ingredients') outlines the other protocols which we use as building blocks.
2. Section 3 ('The protocol') introduces our argument system for BQP.
3. Section 4 ('Soundness of protocol') proves that the argument system introduced in section 3 is sound against quantum polynomial-time provers.
4. Section 5 ('Zero-knowledge property of protocol') proves that the argument system is zero-knowledge (that yes-instance executions can be simulated classically).

Acknowledgments. We thank Zvika Brakerski, Andru Gheorghiu, and Zhengfeng Ji for useful discussions. Thomas Vidick is supported by NSF CAREER Grant CCF-1553477, AFOSR YIP award number FA9550-16-1-0495, MURI Grant FA9550-18-1-0161, a CIFAR Azrieli Global Scholar award, and the IQIM, an NSF Physics Frontiers Center (NSF Grant PHY-1125565) with support of the Gordon and Betty Moore Foundation (GBMF-12500028). Tina Zhang acknowledges support from the Richard G. Brewer Prize and Caltech's Ph11 program.

2 Ingredients

The protocol we present in section 3 combines techniques which were introduced in prior works for the design of protocols to solve related problems. In this section, we outline these protocols in order to introduce notation and groundwork which will prove useful in the remainder of the paper.

2.1 Single-qubit-verifier proof system for QMA ([MF16])

Morimae and Fitzsimons ([MF16]) present a proof system for languages in QMA whose verifier is classical except for a single-qubit quantum register, and which is sound against arbitrary quantum provers. The proof system relies on the QMA-completeness of the 2-local XZ Hamiltonian problem, which is defined as follows.

Definition 2.1 (2-local XZ Hamiltonian (promise) problem).
Input. An input to the problem consists of a tuple $x = (H, a, b)$, where

1. $H = \sum_{s=1}^{S} d_s H_s$ is a Hamiltonian acting on n qubits, each term H_s of which
   (a) has a weight d_s which is a polynomially bounded rational number,
   (b) satisfies $0 \leq H_s \leq I$,
   (c) acts as the identity on all but a maximum of two qubits,
   (d) acts as the tensor product of Pauli observables in $\{\sigma_X, \sigma_Z\}$ on the qubits on which it acts non-trivially.

2. a and b are two real numbers such that
   (a) $a < b$, and
   (b) $b - a = \Omega(\frac{1}{\mathsf{poly}(|x|)})$.

Yes: There exists an n-qubit state σ such that $\langle \sigma, H \rangle \leq a$.[2]

No: For every n-qubit state σ, it holds that $\langle \sigma, H \rangle \geq b$.

Remark 2.2. Given a Hamiltonian H, we call any state σ^* which causes $\langle \sigma^*, H \rangle$ to take its minimum possible value a *ground state* of H, and we refer to the value $\langle \sigma^*, H \rangle$ as the *ground energy* of H.

The following theorem is proven by Biamonte and Love in [BL08, Theorem 2].

Theorem 2.3. *The 2-local XZ Hamiltonian problem is complete for QMA.*

We now describe an amplified version of the protocol presented in [MF16], and give a statement about its completeness and soundness which we will use. (See [MF16] for a more detailed presentation of the unamplified version of this protocol.)

Protocol 2.4 (Amplified variant of the single-qubit-verifier proof system for QMA from [MF16])**.**

Notation. Let L be any language in QMA; let $x \in \{0,1\}^*$ be an input; and let (H, a, b) be the instance of the 2-local XZ Hamiltonian problem to which x reduces.

1. If $x \in L$, the ground energy of H is at most a.
2. if $x \notin L$, the ground energy of H is at least b.
3. $b - a \geq \frac{1}{\mathsf{poly}(|x|)}$.

Let $H = \sum_{s=1}^{S} d_s H_s$, as in Definition 2.1. Define

$$\pi_s = \frac{|d_s|}{\sum_s |d_s|} \, .$$

Parties. The proof system involves

1. A *verifier*, who implements a classical probabilistic polynomial-time procedure with access to a one-qubit quantum register; and
2. A *prover*, who is potentially unbounded, but whose honest behaviour in yes-instances can be implemented in quantum polynomial time.

The verifier and the prover communicate quantumly.

Inputs.

1. Input to the verifier: the Hamiltonian H and the numbers a and b.
2. Input to the prover: the Hamiltonian H, the numbers a and b, and the quantum state $\rho = \sigma^{\otimes m}$, where σ is a ground state of the Hamiltonian H.

Protocol.

[2]The angle brackets $\langle \cdot, \cdot \rangle$ denote an inner product between two operators which is defined as follows: $\langle A, B \rangle = \mathrm{Tr}(A^* B)$ for any $A, B \in \mathrm{L}(\mathcal{X}, \mathcal{Y})$, where the latter denotes the space of linear maps from a Hilbert space $\mathcal{X}$ to a Hilbert space $\mathcal{Y}$.)

1. The verifier selects uniformly random coins $r = (r_1, \dots, r_m)$.
2. For each $j \in \{1, \dots, m\}$, the verifier uses r_j to select a random $s_j \in \{1, \dots, S\}$ according to the distribution D specified as follows:

$$D(s) = \pi_s, \quad \text{for } s \in \{1, \dots, S\} .$$

3. The prover sends a state ρ to the verifier one qubit at a time. (The honest prover sends the state $\sigma^{\otimes m}$ that consists of m copies of the ground state of H.)
4. The verifier measures H_{s_j} for $j = 1, \dots, m$, taking advantage of the fact that—if the prover is honest—it is given m copies of σ. ('Measuring H_{s_j}', in this case, entails performing at most two single-qubit measurements, in either the standard or the Hadamard basis, on qubits in ρ, and then computing the product of the two measurement outcomes.)
5. The verifier initialises a variable COUNT to 0. For each $j \in \{1, \dots, m\}$, if the jth product that it obtained in the previous step was equal to $-\text{sign}(d_j)$, the verifier adds one to COUNT.
6. If $\frac{\text{COUNT}}{m}$ is closer to $\frac{1}{2} - \frac{a}{\sum_s 2|d_s|}$ than to $\frac{1}{2} - \frac{b}{\sum_s 2|d_s|}$, the verifier accepts. Otherwise, it rejects.

Claim 2.5. *Given an instance $x = (H, a, b)$ of the 2-local XZ Hamiltonian problem, there is a polynomial P (depending only on a and b) such that, for any $m = \Omega(P(|x|))$, the following holds. In a yes-instance, the procedure of Protocol 2.4 accepts the state $\rho = \sigma^{\otimes m}$ with probability exponentially close (in $|x|$) to 1. In a no-instance, it accepts with probability exponentially small in $|x|$.*

Proof. Consider the probability (over the choice of r_j and the randomness arising from measurement) that the jth measurement from step 4 of Protocol 2.4 yields $-\text{sign}(d_j)$. Denote this probability by q_j.

As shown in [MNS16, Section IV], it is not hard to verify that

1. when $x \in L$, $q_j \geq \frac{1}{2} - \frac{a}{\sum_s 2|d_s|}$, and
2. when $x \notin L$, $q_j \leq \frac{1}{2} - \frac{b}{\sum_s 2|d_s|}$.

The difference between the two cases is inverse polynomial in the size of the input to the 2-local XZ Hamiltonian problem. It is straightforward to show that, for an appropriate choice of m, this inverse polynomial gap can be amplified to an exponential one: see Appendix B. □

Remark 2.6. It will be useful later to establish at this point that, if the string r from step 1 of Protocol 2.4 is fixed, it is simple to construct a state ρ_r which will pass the challenge determined by r with probability 1. One possible procedure is as follows.

1. For each $j \in 1, \dots, m$:

   Suppose that $H_{s_j} = d_j P_1 P_2$, and that $P_1, P_2 \in \{\sigma_X, \sigma_Z\}$ act on qubits ℓ_1 and ℓ_2, respectively.
   (a) If $-\text{sign}(d_j) = 1$, initialise the $((j-1)n + \ell_1)$th qubit to the $+1$ eigenstate of P_1, and likewise, initialise the $((j-1)n + \ell_2)$th qubit to the $+1$ eigenstate of P_2.
   (b) If $-\text{sign}(d_j) = -1$, initialise the $((j-1)n + \ell_1)$th qubit to the $+1$ eigenstate of P_1, and initialise the $((j-1)n + \ell_2)$th qubit to the -1 eigenstate of P_2.
2. Initialise all remaining qubits to $|0\rangle$.

It is clear that the ρ_r produced by this procedure is a tensor product of $|0\rangle$, $|1\rangle$, $|+\rangle$ and $|-\rangle$ qubits.

2.2 Measurement protocol ([Mah18])

In [Mah18], Mahadev presents a measurement protocol between a quantum prover and a classical verifier which, intuitively, allows the verifier to obtain trustworthy standard and Hadamard basis measurements of the prover's quantum state from purely classical interactions with it. The soundness of the measurement protocol relies upon the security properties of functions that [Mah18] terms *noisy trapdoor claw-free functions* and *trapdoor injective functions*, of which Mahadev provides explicit constructions presuming upon the hardness of LWE. (A high-level summary of these constructions can be found in Appendix A.) Here, we summarise the steps of the protocol, and state the soundness property that it has which we will use.

Protocol 2.7 (Classical-verifier, quantum-prover measurement protocol from [Mah18])**.**

Parties. The proof system involves

1. A *verifier*, which implements a classical probabilistic polynomial-time procedure; and
2. A *prover*, which implements a quantum polynomial-time procedure.

The verifier and the prover communicate classically.

Inputs.

1. Input to the prover: an n-qubit quantum state ρ, whose qubits the verifier will attempt to derive honest measurements of in the standard and Hadamard bases.
2. Input to the verifier:
   (a) A string $h \in \{0,1\}^n$, which represents the bases (standard or Hadamard) in which it will endeavour to measure the qubits of ρ. $h_i = 0$ signifies that the verifier will attempt to obtain measurement outcomes of the ith qubit of ρ in the standard basis, and $h_i = 1$ means that the verifier will attempt to obtain measurement outcomes of the ith qubit of ρ in the Hadamard basis.
   (b) An *extended trapdoor claw-free function family* (ETCFF family), as defined in Section 4 of [Mah18]. The description of an ETCFF family specifies a large number of algorithms, and we do not attempt to enumerate them. Instead, we proceed to describe the verifier's prescribed actions at a level of detail which we believe to be sufficient for our purposes, and refer the reader to [Mah18] for a finer exposition.

Protocol.

1. For each $i \in \{1, \ldots, n\}$ (see 'Inputs' above for the definition of n), the verifier generates an ETCFF function key κ_i using algorithms provided by the ETCFF family, along with a trapdoor τ_{κ_i} for each function, and sends all of the keys κ to the prover. It keeps the trapdoors τ to itself. If $h_i = 0$, the ith key κ_i is a key for a *trapdoor injective* (g) function, and if $h_i = 1$, it is a key for a *noisy trapdoor claw-free* (f) function. Intuitively, the g functions are one-to-one trapdoor one-way functions, and the f functions are *two*-to-one trapdoor collision-resistant hash functions. The keys for f functions and those for g functions are computationally indistinguishable. (For convenience, we will from now on refer to the function specified by κ_i either as f_{κ_i} or as g_{κ_i}. Alternatively, we may refer to it as η_{κ_i} if we do not wish to designate its type.[3]) A brief outline of how these properties are achieved using LWE is given in Appendix A.

   We make two remarks about the functions η_{κ_i} which will become relevant later.
   (a) The functions η_{κ_i} always have domains of the form $\{0,1\} \times \mathcal{X}$, where $\mathcal{X} \subseteq \{0,1\}^w$ for some length parameter w.
   (b) The outputs of both the f and the g functions should be thought of not as strings but as *probability distributions*. The trapdoor τ_{κ_i} inverts the function specified by κ_i in the sense that, given a sample

[3]The letter η has been chosen because it bears some resemblance to the Latin letter h.

y from the distribution $Y = \eta_{\kappa_i}(b\|x)$, along with the trapdoor τ_{κ_i}, it is possible to recover $b\|x$, as well as any other $b'\|x'$ which also maps to Y under η_{κ_i} (should it exist).

Definition 2.8. Suppose that η_{κ_i} is the function specified by κ_i, whose output on each input $b\|x$ in its domain $\{0,1\} \times \mathcal{X}$ is a probability distribution Y. Define a (deterministic) function $\eta^*_{\kappa_i}(b\|x, e)$ which takes as input an $b\|x \in \{0,1\} \times \mathcal{X}$ and a randomness $e \in \mathcal{E}$, for some well-defined finite set $\mathcal{E}$, and returns a sample y_e from the distribution $Y = \eta_{\kappa_i}(b\|x)$.

Definition 2.9. Let η_{κ_i} be the function specified by κ_i, with domain $\{0,1\} \times \mathcal{X}$. Let y be a sample from one of the distributions $Y \in \mathcal{Y}$, where $\mathcal{Y}$ is the range of η_{κ_i}. We call $b\|x \in \{0,1\} \times \mathcal{X}$ a *preimage of* y if y is in the support of the probability distribution $\eta_{\kappa_i}(b\|x)$. (It is guaranteed that the distributions in the range of η_{κ_i} have compact support.)

2. The prover uses the function keys $\kappa_1, \dots, \kappa_n$ to 'commit' to the quantum state of each of the n qubits of ρ, and sends n commitment strings $y_1, \dots, y_n$ to the verifier. We direct the reader to Sections 2.2 and 5.1 of [Mah18] for a high-level, and then a more detailed, description of the commitment procedure, and explanations of how such a procedure will allow the verifier to extract trustworthy standard and Hadamard basis measurements of the qubits of ρ. For convenience, we summarise the procedure without justification here.

   For each i, the prover concatenates to the qubit indexed by i (which we call, following [Mah18], the ith 'committed qubit', and whose state we denote by[4] $|\psi_i\rangle = \gamma_i |0\rangle + \delta_i |1\rangle$) a register (the 'preimage register') containing a uniform superposition over all $x \in \mathcal{X}$ (where $\{0,1\} \times \mathcal{X}$ is the domain of the function η_{κ_i}). It also concatenates to it a register containing a superposition over all $e \in \mathcal{E}$, with $\mathcal{E}$ defined as it is in Definition 2.8. It then applies the function $\eta^*_{\kappa_i}$ (see Definition 2.8) in superposition to $\sum_x |\psi_i\rangle |x\rangle |e\rangle$, and collects the outputs in a new register (the 'output register'), obtaining the quantum state

$$\sum_{x \in \{0,1\}^w} \gamma_i |0\rangle |x\rangle |e\rangle |\eta^*_{\kappa_i}(0\|x, e)\rangle + \delta_i |1\rangle |x\rangle |e\rangle |\eta^*_{\kappa_i}(1\|x, e)\rangle \ .$$

   Finally, the prover measures the output register, obtains a sample y_i, and sends y_i to the verifier as its ith commitment string.

3. The verifier then chooses at random to run either a *test round* or a *Hadamard round*. Each option is chosen with probability $\frac{1}{2}$.
   (a) If a test round is chosen, the verifier requests standard basis measurements of each committed qubit $|\psi_i\rangle$ and its associated preimage register (recall that, in the previous step, only the output registers were measured), and receives measurement results β_i, x_i for $i \in \{1, \dots, n\}$ from the prover. It then checks, for each $i \in \{1, \dots, n\}$, whether the ith measurement result is a preimage of y_i. (See Definition 2.9 for a definition of the phrase 'preimage of y'.) More precisely: if $h_i = 0$, the verifier accepts iff y_i is in the support of $g_{\kappa_i}(\beta_i\|x_i)$ (recalling that $g_{\kappa_i}(\beta_i\|x_i)$ is a probability distribution); and if $h_i = 1$, the verifier accepts iff y_i is in the support of $f_{\kappa_i}(\beta_i\|x_i)$.
   (b) If a Hadamard round is chosen, the verifier requests Hadamard basis measurements of each committed qubit and its associated preimage register, and receives β_i, d_i for $i \in \{1, \dots, n\}$ from the prover. (We note, for future reference, that the distribution of the strings d_i sent by the honest prover is the uniform distribution over $\{0,1\}^w$.) It then obtains the measurement results it desires using the procedure outlined in Section 2.2 of [Mah18]:
      i. If $h_i = 0$, the verifier ignores the measurement results β_i, d_i, and uses the trapdoor τ_{κ_i} to invert y_i. The inversion procedure outputs a preimage (m_i, x_{m_i, y_i}); the verifier stores m_i as the standard-basis measurement outcome. (If the inverse does not exist, the verifier stores a

[4]Strictly speaking, the state of the ith commited qubit may not be a pure state, but we ignore this fact for simplicity of presentation.

random bit as m_i and rejects. The specification that the verifier stores a random bit is made so that the n-bit string of measurement results, m, is well-defined even when the verifier rejects.)

ii. If $h_i = 1$, the verifier computes the two preimages $x_{0,i}$ and $x_{1,i}$ to y_i under f_{κ_i} using the trapdoor τ_{κ_i}. It then stores $\beta_i \oplus d_i \cdot (x_{0,i} \oplus x_{1,i})$ as the Hadamard basis measurement outcome. (Once again, if either inverse does not exist, the verifier stores a random bit as m_i and rejects.)

The soundness property of this measurement protocol (Claim 7.1 in [Mah18]) is stated below. Before we state it, we briefly introduce the notation used in [Mah18]. (For a more detailed exposition, we refer the reader to Section 5.3 of [Mah18].) $D_{\mathbb{P},h}$ refers to the distribution over measurement results $m \in \{0,1\}^n$ that the verifier obtains when it executes a Hadamard round with the prover labelled $\mathbb{P}$ on the basis choice h. $D^C_{\mathbb{P},h}$ is the same distribution, but conditioned on the verifier accepting (in a Hadamard round). $D_{\xi,h}$ is the distribution over measurement outcomes in $\{0,1\}^n$ that would result from directly measuring the quantum state ξ in the bases determined by h. $p_{h,T}$ and $p_{h,H}$ are defined so that the verifier's probability of accepting (on basis choice h) in a test and a Hadamard round, respectively, are $1-p_{h,T}$ and $1-p_{h,H}$. $\|\cdot\|_{TV}$ denotes the total variation norm, and $A \approx_c B$ indicates that two distributions A and B are (quantum) computationally indistinguishable.

Claim 2.10. *Assume that the Learning With Errors problem (with the same choices of parameters as those made in [Mah18, Section 9]) is quantum computationally intractable. Then, for any arbitrary quantum polynomial-time prover $\mathbb{P}$ who executes the measurement protocol (Protocol 2.7) with the honest verifier V, there exists a quantum state ξ, a prover $\mathbb{P}'$ and a negligible function μ such that*

$$\|D^C_{\mathbb{P},h} - D_{\mathbb{P}',h}\|_{TV} \leq \sqrt{p_{h,T}} + p_{h,H} + \mu \quad \textit{and}$$
$$D_{\mathbb{P}',h} \approx_c D_{\xi,h} .$$

2.3 Zero-knowledge proof system for QMA ([BJSW16])

In [BJSW16], Broadbent, Ji, Song and Watrous describe a protocol involving a quantum polynomial-time verifier and an unbounded prover, interacting quantumly, which constitutes a zero-knowledge proof system for languages in QMA. (Although it is sound against arbitrary provers, the system in fact only requires an honest prover to perform quantum polynomial-time computations.) We summarise the steps of their protocol below. For details and fuller explanations, we refer the reader to [BJSW16, Section 3].

Protocol 2.11 (Zero-knowledge proof system for QMA from [BJSW16])**.**

Notation. Let L be any language in QMA. For a definition of the *k-local Clifford Hamiltonian problem*, see [BJSW16, Section 2]. The k-local Clifford Hamiltonian problem is QMA-complete for $k = 5$; therefore, for all possible inputs x, there exists a 5-local Clifford Hamiltonian H (which can be computed efficiently from x) whose terms are all operators of the form $C^* |0^k\rangle \langle 0^k| C$ for some Clifford operator C, and such that

1. if $x \in L$, the ground energy of H is $\leq 2^{-p}$,
2. if $x \notin L$, the ground energy of H is $\geq \frac{1}{q}$,

for some positive integers p and q.

Parties. The proof system involves

1. A *verifier*, who implements a quantum polynomial-time procedure;
2. A *prover*, who is unbounded, but who is only required by the protocol to implement a quantum polynomial-time procedure.

The verifier and the prover communicate quantumly.

Inputs.

1. Input to the verifier:
   (a) The Hamiltonian H.
   (b) A quantum computationally concealing, perfectly binding (classical) commitment protocol.
   (c) A proof system for NP sound against arbitrary quantum provers.
2. Input to the prover:
   (a) The Hamiltonian H.
   (b) The n-qubit quantum state ρ, where ρ is a ground state of the Hamiltonian H.
   (c) A quantum computationally concealing, perfectly binding (classical) commitment protocol.
   (d) A proof system for NP sound against arbitrary quantum provers.

Protocol.

1. *The prover's encoding step.* The prover firstly applies a concatenated Steane code (which maps every one qubit to N qubits) to each qubit in ρ. (For details on the concatenated Steane code, see [BJSW16, Appendix A.6]. It will be important to Broadbent et al.'s purposes—and ours, in Section 3—that this code admits transversal applications of Clifford operations.) It then executes the following steps:
   (a) Concatenate N trap qubits to the end of each logical qubit (alternatively, to the end of each N-qubit block) in the result of applying the concatenated Steane code to ρ. Each trap qubit is initialised uniformly at random to one of $|0\rangle, |+\rangle, |+_y\rangle$.[5] Denote the string that records the choices of trap qubits for all n logical qubits by $t = t_1, \ldots, t_n$.
   (b) Permute each $2N$-tuple of qubits in the result of (a) according to a uniformly random permutation $\pi \in S_{2N}$. (Note that the same permutation π is applied to every $2N$-tuple.)
   (c) Apply a Pauli one-time pad $X^a Z^b$, for uniformly random $a, b \in \{0,1\}^{2nN}$, to the entire $2nN$-qubit state.

   Remark 2.12. The number N functions as a security parameter, both for this protocol and for the protocol that we construct in section 3. Accordingly, we assume that N is polynomial in the size of the verifier's input.

   The prover's encoding applied to ρ is denoted by $E(\rho)$, and the procedure E is fully determined by the encoding key (t, π, a, b) which the prover chose to use. At this point, the prover sends the state $E(\rho)$ to the verifier, along with a commitment (using some perfectly binding, computationally concealing classical commitment protocol) to the tuple (π, a, b). (A commitment to the sequence of trap qubits t is unnecessary because, in a sense, the trap qubits exist only to check the verifier.) Let the prover's commitment string be denoted z.
2. *Coin-flipping protocol.* The prover and the verifier execute a coin-flipping protocol, choosing a string r of fixed length uniformly at random. This random string r determines a local Hamiltonian term $H_r = C_r^* |0^k\rangle \langle 0^k| C_r$ that is to be tested. (This step can be implemented, of course, using the same classical commitment protocol that the prover employed in the previous step.)
3. *Verifier's challenge.* The verifier applies the Clifford C_r transversally to the qubits on which the k-local Hamiltonian term H_r acts nontrivially, and measures them in the standard basis. It then sends the measurement results $u_{i_1}, \ldots, u_{i_k}$ which it obtained to the prover. (Each u_i is a $2N$-bit string, and $i_1, \ldots, i_k$ are the indices of the logical qubits on which the term H_r acts nontrivially.)

[5] $|+_y\rangle$ here refers to the state $\frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$.

4. *Prover's response.* The prover receives the verifier's measurement results u, and firstly checks whether they cause a predicate $Q(t, \pi, a, b, r, u)$ to be satisfied. (We will explain the predicate Q in more detail shortly. Intuitively, Q is satisfied if and only if both verifier *and* prover behaved honestly.) If Q is not satisfied, the prover aborts, causing the verifier to reject. If Q is satisfied, then the prover proves to the verifier, using an NP zero-knowledge protocol, that there exists randomness s_p and an encoding key (t, π, a, b) such that $z = \mathsf{commit}((\pi, a, b), s_p)$ and $Q(t, \pi, a, b, r, u) = 1$.

   We now describe the predicate Q in precise terms. For convenience, Broadbent et al. define a predicate R_r, which represents the prover's check after it reverses the effects on u of the one-time pad $X^a Z^b$, and then proceed to define Q in terms of R_r. Since we will shortly have cause to alter the definition of R_r, we quote the definition of R_r used in [BJSW16] for comparison. (Note that we have altered their notation slightly: the strings that they call y_i we have called q_i, to avoid a conflict.)

   Definition 2.13. *Definition of R_r.*

   Let $u_{i_1}, \ldots, u_{i_k}$ be the measurement results that the verifier sent to the prover in step 3. For each $i \in \{i_1, \ldots, i_k\}$, define strings q_i, z_i in $\{0,1\}^N$ such that $\pi(q_i \| z_i) = u_i$ (alternatively: $\pi^{-1}(u_i) = q_i \| z_i$). The predicate $R_r(t, \pi, u)$ takes the value 1 if and only if the following two conditions are met:

   (a) $q_i \in \mathcal{D}_N$ for every $i \in \{i_1, \ldots, i_k\}$, and $q_i \in \mathcal{D}_N^1$ for at least one index $i \in \{i_1, \ldots, i_k\}$. ($\mathcal{D}_N = \mathcal{D}_N^0 \cup \mathcal{D}_N^1$ is the set of all valid classical N-bit codewords of the concatenated Steane code. We note, as an aside, that $\mathcal{D}_N^0$ and $\mathcal{D}_N^1$ are both sets of size 8^t, where t is an even positive integer such that $7^t = N$; as such, $\mathcal{D}_N$ is polynomially sized.)

   (b) $\langle z_{i_1} \cdots z_{i_k} | \, C_r^{\otimes n} \, | t_{i_1} \cdots t_{i_k} \rangle \neq 0$.

   Now we define the predicate $Q(t, \pi, a, b, r, u)$ in terms of R_r:

   Definition 2.14. *Definition of Q.*

   Let $c_1, \ldots, c_n$, $d_1, \ldots, d_n \in \{0,1\}^{2N}$ be the unique strings such that

$$C_r^{\otimes 2N}(X^{a_1} Z^{b_1} \otimes \cdots \otimes X^{a_n} Z^{b_n}) = \alpha(X^{c_1} Z^{d_1} \otimes \cdots \otimes X^{c_n} Z^{d_n}) C_r^{\otimes 2N}$$

   for some $\alpha \in \{1, i, -1, -i\}$. (It is possible to efficiently compute $c = c_1, \ldots, c_n$ and $d = d_1, \ldots, d_n$ given a, b and C_r.) The predicate Q is then defined by

$$Q(t, \pi, a, b, r, u) = R_r(t, \pi, u \oplus c_{i_1} \cdots c_{i_k}) \, .$$

2.4 Replacing Clifford verification with XZ verification in Protocol 2.11

The authors of [BJSW16] introduce a zero-knowledge proof system which allows the verifier to determine whether the prover holds a state that has sufficiently low energy with respect to a k-local Clifford Hamiltonian (see Section 2 of [BJSW16]). In this section, we modify their proof system so that it applies to an input encoded as an instance of the XZ local Hamiltonian problem (Definition 2.1) rather than as an instance of the Clifford Hamiltonian problem.

Before we introduce our modifications, we explain why it is necessary in the first place to alter the proof system presented in [BJSW16]. Modulo the encoding E which the prover applies to its state in Protocol 2.11, the quantum verifier from the same protocol is required to perform a projective measurement of the form $\{\Pi = C^* |0^k\rangle \langle 0^k| C, \mathrm{Id} - \Pi\}$ of the state that the prover sends it (where C is a Clifford unitary acting on k qubits) and reject if it obtains the first of the two possible outcomes. Due to the properties of Clifford unitaries, this action is equivalent to measuring k commuting k-qubit Pauli observables $C^* Z_i C$ for $i \in \{1, \ldots, k\}$ (where Z_i is a Pauli σ_Z observable acting on the ith qubit), and rejecting if all of said measurements result in the outcome $+1$.

Our goal is to replace the quantum component of the verifier's actions in Protocol 2.11—a component which, fortunately, consists entirely of performing the projective measurement just described—with the measurement protocol introduced in [Mah18] (summarized as Protocol 2.7). Unfortunately, the latter protocol 1. only allows for standard and Hadamard basis measurements, and 2. does not accommodate a verifier who wishes to perform multiple successive measurements on the same qubit: for each qubit that the verifier wants to measure, it must decide on a measurement basis (standard or Hadamard) prior to the execution of the protocol, and once made its choices are fixed for the duration of its interaction with the prover. This allows the verifier to, for example, obtain the outcome of a measurement of the observable $C^* Z_i C$ for some *particular* i, by requesting measurement outcomes of all k qubits in the appropriate basis and taking the product of the outcomes obtained. However, it is not obvious how the same verifier could request the outcome of measuring a *k-tuple* of commuting Pauli observables which all act on the same k qubits.

To circumvent this technical issue, we replace the Clifford Hamiltonian problem used in [BJSW16] with the QMA-complete XZ Hamiltonian problem. The advantage of this modification is that it becomes straightforward to implement the required energy measurements using the measurement protocol from [Mah18]. Unfortunately, in order to make the change, we sacrifice the perfect completeness which is a property of the proof system in [BJSW16], and also require that the verifier's measurements act on a linear, rather than a constant, number of qubits with respect to the size of the problem input.

A different potentially viable modification to the proof system of [BJSW16] is as follows. Instead of replacing Clifford Hamiltonian verification with XZ Hamiltonian verification, we could also repeat the original Clifford-Hamiltonian-based protocol a polynomial number of times. In such a scheme, the honest prover would hold m copies of the witness state (as it does in Protocol 2.4). The verifier, meanwhile, would firstly choose a random term $C_r^* \left|0^k\right\rangle \left\langle 0^k\right| C_r$ from the Clifford Hamiltonian, and then select m random Pauli observables of the form $C_r^* Z_i C_r$—where C_r is the particular C_r which it picked—to measure. (For each repetition, i would be chosen independently and uniformly at random from the set $\{1, \ldots, k\}$.) The verifier would accept if and only if the number of times it obtains -1 from said Pauli measurements is at least $\frac{m}{2k}$. This approach is very similar to the approach we take for XZ Hamiltonians (which we explain below), and in particular also fails to preserve the perfect completeness of the original protocol in [BJSW16]. For simplicity, we choose the XZ approach. We now introduce the alterations which are necessary in order to make it viable.

Firstly, we require that the honest prover possesses polynomially many copies of the witness state σ, instead of one. We do this because we want the honest verifier to accept the honest prover with probability exponentially close to 1, which is not naturally true in the verification procedure for 2-local XZ Hamiltonians presented by Morimae and Fitzsimons in [MF16], but which is true in our amplified variant, Protocol 2.4. Secondly, we need to modify the verifier's conditions for acceptance. In [BJSW16], as we have mentioned, these conditions are represented by a predicate Q (that in turn evaluates a predicate R_r; see Definitions 2.13 and 2.14).

We now describe our alternative proof system for QMA, and claim that it is zero-knowledge. Because the protocol is very similar to the protocol from [BJSW16], this can be seen by following the proof of zero-knowledge in [BJSW16], and noting where our deviations require modifications to the reasoning. On the other hand, we do not argue that the proof system is complete and sound, as we do not need to make explicit use of these properties. (Intuitively, however, the completeness and the soundness of the proof system follow from those of Protocol 2.4, and the soundness of the latter is a property which we will use.)

Protocol 2.15 (Alternative proof system for QMA)**.**

Notation. Refer to notation section of Protocol 2.4.

Parties. The proof system involves

1. A *verifier*, who implements a quantum polynomial-time procedure;
2. A *prover*, who is unbounded, but who is only required by the protocol to implement a quantum

polynomial-time procedure.

The verifier and the prover communicate quantumly.

Inputs.

1. Input to the verifier:
   (a) The Hamiltonian H, and the numbers a and b.
   (b) A quantum computationally concealing, perfectly binding (classical) commitment protocol.
   (c) A proof system for NP sound against arbitrary quantum provers.
2. Input to the prover:
   (a) The Hamiltonian H, and the numbers a and b.
   (b) The n-qubit quantum state $\rho = \sigma^{\otimes n}$, where σ is the ground state of the Hamiltonian H.
   (c) A quantum computationally concealing, perfectly binding (classical) commitment protocol.
   (d) A proof system for NP sound against arbitrary quantum provers.

Protocol.

1. *Prover's encoding step:* The same as the prover's encoding step in Protocol 2.11, except that $t \in \{0, +\}^N$ rather than $\{0, +, +_y\}^N$. (This change will be justified in the proof of Lemma 2.18.)
2. *Coin flipping protocol:* Unmodified from Protocol 2.11, except that $r = (r_1, \ldots, r_m)$ represents the choice of m terms from the 2-local XZ Hamiltonian H (with the choices being made as described in step 2 of Protocol 2.4) instead of a random Clifford. Note that r determines the indices of the $2m$ logical qubits which the verifier will measure in step 3.
3. *Verifier's challenge:* The same as the verifier's challenge in Protocol 2.11, except that the verifier now applies U_r transversally instead of C_r.
4. *Prover's response:* The same as Protocol 2.11 (but note that the predicate Q, which the prover checks and then proves is satisfied, is the Q described in Definition 2.17 below).

Definition 2.16 (Redefinition of R_r)**.** Let $i_1, \ldots, i_{2m}$ be the indices of the logical qubits which were chosen for measurement in step 2 of Protocol 2.15, ordered by their corresponding js (so that i_1 and i_2 are the qubits that were measured in order to determine whether H_{s_1} was satisfied, and so on). Let $u_{i_1}, \ldots, u_{i_{2m}}$ be the $2N$-bit strings which the verifier claims are the classical states that remained after said measurements were performed, and for each $i \in \{i_1, \ldots, i_{2m}\}$, define N-bit strings q_i, z_i such that $\pi(q_i || z_i) = u_i$ (alternatively: $\pi^{-1}(u_i) = q_i || z_i$). In Protocol 2.15, the predicate $R_r(t, \pi, u)$ takes the value 1 if and only if the following conditions are met:

1. $q_i \in \mathcal{D}_N$ for every $i \in \{i_1, \ldots, i_{2m}\}$.
2. The number $\frac{\textsc{Count}}{m}$ (where $\textsc{Count}$ is obtained by executing the following procedure) is closer to $\frac{1}{2} - \frac{a}{\sum_s 2|d_s|}$ than to $\frac{1}{2} - \frac{b}{\sum_s 2|d_s|}$.
   (a) Initialise $\textsc{Count}$ to 0.
   (b) For each $j \in \{1, \ldots, m\}$: Suppose that $H_{s_j} = d_j P_1 P_2$, for some $P_1, P_2 \in \{\sigma_X, \sigma_Z\}$. The tuple $(P_1, u_{2j-1}, P_2, u_{2j})$ determines a 'logical' measurement result that could equally have been obtained by measuring $H_{r_j}\sigma$, where σ is the unencoded witness state. We denote this measurement result by λ. If $\lambda = -\text{sign}(d_j)$, add one to $\textsc{Count}$.
   (c) Let U_r be the circuit obtained from the following procedure:
      i. For each $j \in \{1, \ldots, m\}$, replace any σ_Xs in the term H_{s_j} with H (Hadamard) gates, and replace any σ_Zs in H_{s_j} with I. (For example, if $H_{r_j} = \pi_j \sigma_{X,\ell_1} \sigma_{Z,\ell_2}$, where the second subscript denotes the index of the qubit on which the observable in question acts, then $U_j = H_{\ell_1} I_{\ell_2}$,

where the subscripts ℓ_1 and ℓ_2 once again the denote the indices of the qubits on which the gates H and I act.)

ii. Apply U_j to the qubits indexed $(j-1)n+1$ through jn.

It must then be the case that $\langle z_{i_1} \cdots z_{i_{2m}} | U_r^{\otimes N} | t_{i_1} \cdots t_{i_{2m}} \rangle \neq 0$ (where each t_i is an N-bit string that represents the pattern of trap qubits which was concatenated to the ith logical qubit during step 1 of Protocol 2.15).

Definition 2.17 (Redefinition of Q)**.** Let $c_1, \ldots, c_n,\ d_1, \ldots, d_n \in \{0,1\}^{2N}$ be the unique strings such that

$$U_r^{\otimes 2N}(X^{a_1}Z^{b_1} \otimes \cdots \otimes X^{a_n}Z^{b_n}) = \alpha(X^{c_1}Z^{d_1} \otimes \cdots \otimes X^{c_n}Z^{d_n})U_r^{\otimes 2N}$$

for some $\alpha \in \{1, i, -1, -i\}$. (It is possible to efficiently compute $c = c_1, \ldots, c_n$ and $d = d_1, \ldots, d_n$ given a, b and U_r. In particular, recalling that U_r is a tensor product of H and I gates, we have that $c_i = a_i$ and $d_i = b_i$ for all i such that the ith gate in $U_r^{\otimes 2N}$ is I, and $c_i = b_i$, $d_i = a_i$ for all i such that the ith gate in $U_r^{\otimes 2N}$ is H.) The predicate Q is then defined by

$$Q(t, \pi, a, b, r, u) = R_r(t, \pi, u \oplus c_{i_1} \cdots c_{i_k}) \ ,$$

where R_r is as in Definition 2.16.

Lemma 2.18. *The modified proof system for QMA in Protocol 2.15 is computationally zero-knowledge for quantum polynomial-time verifiers.*

Proof. We follow the argument from [BJSW16, Section 5]. Steps 1 to 3 only make use of the security of the coin-flipping protocol, the security of the commitment scheme, and the zero-knowledge properties of the NP proof system, none of which we have modified. Step 4 replaces the real witness state ρ with a simulated witness ρ_r that is guaranteed to pass the challenge indexed by r; this we can do also (see Remark 2.6). Step 5 uses the Pauli one-time-pad to twirl the cheating verifier, presuming that the honest verifier would have applied a Clifford term indexed by r before measuring. We note that, since U_r is a Clifford, the same reasoning applies to our modified proof system.

Finally, using the fact that the Pauli twirl of step 5 restricts the cheating verifier to XOR attacks, step 6 from [BJSW16, Section 5] proves the following statement: if the difference $|p_0 - p_1|$ is negligible (where p_0 and p_1 are the probabilities that ρ and ρ_r respectively pass the verifier's test in an honest prover-verifier interaction indexed by r), then the channels Ψ_0 and Ψ_1 implemented by the cheating verifier in each case are also quantum computationally indistinguishable. It follows from this statement that the protocol is zero-knowledge, since, in an honest verifier-prover interaction indexed by r, ρ_r would pass with probability 1, and ρ would pass with probability $1 - \mathsf{negl}(N)$. (This latter statement is true both in their original and in our modified protocol.) The argument presented in [BJSW16] considers two exclusive cases: the case when $|v|_1 < K$, where v is the string that the cheating verifier XORs to the measurement results, $|v|_1$ is the Hamming weight of that string, and K is the minimum Hamming weight of a nonzero codeword in $\mathcal{D}_N$; and the case when $|v|_1 \geq K$. The analysis in the former case translates to Protocol 2.15 without modification, but in the latter case it needs slight adjustment.

In order to address the case when $|v|_1 \geq K$, Broadbent et al. use a lemma which—informally—states that the action of a Clifford on k qubits, each of which is initialised uniformly at random to one of $|0\rangle, |+\rangle$, or $|+\rangle_y$, has at least a 3^{-k} chance of leaving at least one out of k qubits in a standard basis state. We may hesitate to replicate their reasoning directly, because our k (the number of qubits on which our Hamiltonian acts) is not a constant. While it is possible that a mild modification suffices to overcome this problem, we note that in our case there is a simpler argument for an analogous conclusion: since U_r is a tensor product of only H gates and I gates, it is straightforward to see that, if each of the $2m$ qubits on which it acts is initialised either to $|0\rangle$ or to $|+\rangle$, then 1) each of the $2m$ qubits has exactly a 50% chance of being left in a standard basis state, and 2) the states of these $2m$ qubits are independent.

Now we consider the situation where a string $v = v_1\, v_2\, \cdots\, v_{2m}$, of length $4mN$ and of Hamming weight at least K, is permuted ('permuted', here, means that $\pi \in S_{2N}$ is applied to each v_i individually) and then XORed to the result of measuring $4mN$ qubits ($2m$ blocks of $2N$ qubits each) in the standard basis after U_r has been transversally applied to those qubits. It is straightforward to see, by an application of the pigeonhole principle, that there must be at least one v_i whose Hamming weight is $\geq \frac{K}{2m}$. Consider the result of XORing this v_i to its corresponding block of measured qubits. Half of the $2N$ qubits in that block would originally have been encoding qubits, and half would have been trap qubits; half again of the latter, then, would have been trap qubits left in a standard basis state by the transversal action of U_r. As such, the probability that none of the 1-bits of v_i are permuted into positions which are occupied by the latter kind of qubit is $(\frac{3}{4})^{-\frac{K}{2m}}$, which is negligibly small as long as K is made to be a higher-order polynomial in N than $2m$ is. The remainder of the argument in [BJSW16, Section 5] follows directly. □

3 The protocol

In this section, we present our construction of a zero-knowledge argument system for languages in BQP. The argument system can be implemented with a verifier that runs in probabilistic polynomial time and a prover that runs in quantum polynomial time, and is sound under the following assumptions:

Assumptions 3.1.

1. The Learning With Errors problem (LWE) [Reg09] is quantum computationally intractable. (Specifically, we make the same asssumption about the hardness of LWE that is made in [Mah18, Section 9] in order to prove the soundness of the measurement protocol.)
2. There exists a commitment scheme ($\mathsf{gen}, \mathsf{initiate}, \mathsf{commit}, \mathsf{reveal}, \mathsf{verify}$) of the form described in Appendix C that is unconditionally binding and quantum computationally concealing. (This assumption is necessary to the soundness of the proof system presented in [BJSW16].)

Remark 3.2. Our argument system also allows a classical probabilistic polynomial-time verifier and a quantum polynomial-time prover to verify that any problem instance x belongs to any particular language $\mathcal{L} \in$ QMA, provided that the prover has access to polynomially many copies of a valid quantum witness for an instance of the 2-local XZ local Hamiltonian problem to which x is mapped by the reduction implicit in Theorem 2.3.

The following exposition of our protocol relies on definitions from Section 2, and we encourage the reader to read that section prior to approaching this one. We also direct the reader to Figures **??** and **??** for diagrams that chart the protocol's structure.

Protocol 3.3. *Zero-knowledge, classical-verifier argument system for BQP.*

Notation. Let L be any language in BQP (or QMA), and let $(H = \sum_{s=1}^{S} d_s H_s, a, b)$ be an instance of the 2-local XZ Hamiltonian problem to which L can be reduced (see Definition 2.1 and Theorem 2.3). Define

$$\pi_s = \frac{|d_s|}{\sum_s |d_s|}\ .$$

Following [BJSW16], we take the security parameter for this protocol to be N, the number of qubits in which the concatenated Steane code used during the encoding step of the protocol (step 1) encodes each logical qubit. We assume, accordingly, that N is polynomial in the size of the problem instance x.

Parties.

The protocol involves

1. A *verifier*, which runs in classical probabilistic polynomial time;
2. A *prover*, which runs in quantum polynomial time.

Inputs. The protocol requires the following primitives:

- A perfectly binding, quantum computationally concealing commitment protocol (gen, $\mathsf{initiate}$, commit, reveal, verify) (which will be used twice: once for the prover's commitment in step 2, and then again for the coin-flipping protocol in step 3). We assume that this commitment protocol is of the form described in Appendix C.
- A zero-knowledge proof system for NP.
- An extended trapdoor claw-free function family (ETCFF family), as defined in [Mah18]. (Note that we fall short of using the ETCFF family as a black box: for the trapdoor check of step 8, we rely on the specific properties of the LWE-based construction of an ETCFF family that [Mah18] provides. See Appendix A for details.)

Apart from the above cryptographic primitives, we assume that the verifier and the prover also receive the following inputs.

1. Input to the verifier: the Hamiltonian H and the numbers a and b.
2. Input to the prover: the Hamiltonian H, the numbers a and b, and the quantum state $\rho = \sigma^{\otimes m}$, where σ is a ground state of the Hamiltonian H.

Protocol.

1. *The prover encodes the witness.* The prover encodes the quantum witness ρ by applying the following steps:
   (a) Apply concatenated Steane code
   (b) Concatenate trap qubits $|t\rangle$
   (c) Apply a random permutation π
   (d) Apply a Pauli one-time-pad $X^a Z^b$

   The encoding process here is the same as that from step 1 of Protocol 2.15; we direct the reader to Protocol 2.15, and the Protocol 2.11 to which it refers, for a more detailed explanation of the steps. Denote the application of the prover's encoding to the state ρ by $E(\rho)$.
2. *The prover commits to its encoding keys.* The prover commits to the strings (π, a, b) from the previous step, using randomness s_p. Call the prover's commitment string z, so that $z = \mathsf{commit}((\pi, a, b), s_p)$.
3. *The verifier and the prover execute the first half of a two-stage coin-flipping protocol.*[6] The verifier commits to r_v, its part of the random string that will be used to determine which random terms in the Hamiltonian H it will check in subsequent stages of the protocol. Let $c = \mathsf{commit}(r_v, s_v)$. The prover sends the verifier r_p, which is its own part of the random string. The random terms will be determined by $r = r_v \oplus r_p$. (r is used to determine these terms in the same way that r is used in Protocol 2.4.)
4. *The verifier initiates the measurement protocol. (Refer to Protocol 2.7 for an outline of the steps in said measurement protocol.)* The verifier chooses the measurement bases $h = h_1 \cdots h_{2nN}$ in which it wishes to measure the state $E(\rho)$. $2kN$ out of the $2nN$ bits of h—corresponding to k logical qubits—are chosen so that the verifier can determine whether σ satisfies the Hamiltonian terms specified by $r = r_v \oplus r_p$. In our particular case, $k = 2m$, where m is the number of Hamiltonian terms that the verifier will check are satisfied. For the remaining qubits i, the verifier sets h_i to 0. The verifier sends the function keys $\kappa = \kappa_1, \ldots, \kappa_{2nN}$ to the prover.

[6]We need to execute the coin-flipping protocol in two stages because, in our (classical-verifier) protocol, the prover cannot physically send the quantum state $E(\rho)$ to its verifier before the random string r is decided, as the prover of Protocol 2.11 does. If we allow our prover to know r at the time when it performs measurements on the witness ρ, it will trivially be able to cheat.

5. *The prover commits to its encoded witness state, as per the measurement protocol.* The prover commits to the quantum state $E(\rho)$ by concatenating a preimage register to each qubit in $E(\rho)$, applying the functions specified by $\kappa_1, \ldots, \kappa_{2nN}$ in superposition as Protocol 2.7 describes, measuring the resulting output superpositions, and sending the outcomes $y_1, \ldots, y_{2nN}$ to the verifier.

6. *The verifier chooses at random to run either a test round or a Hadamard round.* Each option is chosen with probability $\frac{1}{2}$.
   (a) If a test round is chosen, the verifier requests standard basis measurements of each committed qubit $|\psi_i\rangle$ in $E(\rho)$ and its associated preimage register, and receives measurement results β_i, x_i for $i \in \{1, \ldots, 2nN\}$ from the prover. It then checks, for each $i \in \{1, \ldots, 2nN\}$, whether the ith measurement result is a preimage of y_i. (See Definition 2.9 for a definition of the phrase 'preimage of y'.) More precisely: if $h_i = 0$, the verifier accepts iff y_i is in the support of $g_{\kappa_i}(\beta_i \| x_i)$ (recalling that $g_{\kappa_i}(\beta_i \| x_i)$ is a probability distribution); and if $h_i = 1$, the verifier accepts iff y_i is in the support of $f_{\kappa_i}(\beta_i \| x_i)$.
   (b) If a Hadamard round is chosen, the verifier requests Hadamard basis measurements of each committed qubit and its associated preimage register, and receives β_i, d_i for $i \in \{1, \ldots, 2nN\}$ from the prover. It then obtains the measurement results it desires using the procedure outlined in Section 2.2 of [Mah18]:
      i. If $h_i = 0$, the verifier ignores the measurement results β_i, d_i, and uses the trapdoor τ_{κ_i} to invert y_i. The inversion procedure outputs a preimage (m_i, x_{m_i, y_i}); the verifier stores m_i as the standard-basis measurement outcome. (If the inverse does not exist, the verifier stores a random bit as m_i and rejects.)
      ii. If $h_i = 1$, the verifier computes the two preimages $x_{0,i}$ and $x_{1,i}$ to y_i under f_{κ_i} using the trapdoor τ_{κ_i}. It then stores $\beta_i \oplus d_i \cdot (x_{0,i} \oplus x_{1,i})$ as the Hadamard basis measurement outcome. (Once again, if either inverse does not exist, the verifier stores a random bit as m_i and rejects.)

7. *The verifier opens its commitment, and sends the prover its measurement outcomes and function trapdoors.* If a test round was chosen in the previous step, the verifier now accepts the prover directly, provided that the preimage test was satisfied for all i. (If not all the preimage tests passed, the verifier directly rejects the prover.) If a Hadamard round was chosen, and the verifier instead opens its commitment to r_v and sends its measurement outcomes $m = m_1, \ldots, m_{2nN}$ to the prover, along with its trapdoors $\tau_{\kappa_1}, \ldots, \tau_{\kappa_{2nN}}$.

8. *The prover checks that the verifier's measurement outcomes and trapdoors were consistent; it then proves that the verifier's measurement outcomes were consistent.* The prover checks that $\tau_{\kappa_1}, \ldots, \tau_{\kappa_{2nN}}$ are indeed trapdoors for the functions specified by $\kappa_1, \ldots, \kappa_{2nN}$, by checking that, for each y_i, the trapdoor τ_{κ_i} returns all the preimages to y_i under the function specified by κ_i, and no others[7]. (See Definition 2.9 for a definition of the phrase 'preimages to y'.) It also defines $u = u_{i_1} \cdots u_{i_{2m}}$ (each u_i is $2N$ bits long) $= m_{\ell_1} \cdots m_{\ell_{4mN}}$, where $\ell_1, \ldots, \ell_{4mN}$ are the indices of the qubits on which $U_r^{\otimes 2N}$ acts nontrivially, and checks that u causes the predicate $Q(t, \pi, a, b, r, u)$ to be satisfied. (The Q we refer to here is the Q of Definition 2.17. We define U_r in the same way that U_r was defined in Definition 2.16.) If either of these tests fails, the prover aborts. If both tests pass, then the prover proves, using an NP zero-knowledge proof system,[8] that the verifier's outcomes are consistent in the following sense:

   The verifier's outcomes u are consistent if there exists a string s_p and an encoding key (t, π, a, b) such that $z = \mathsf{commit}((\pi, a, b), s_p)$ and $Q(t, \pi, a, b, r, u) = 1$.

[7]We describe an efficient procedure for administering this test in Appendix A, and give a sketch of a proof that it delivers the guarantee we describe.

[8]It was shown in [Wat09] that the second item in Assumptions 3.1 suffices to guarantee the existence of a proof system for languages in NP that is zero-knowledge against quantum polynomial-time verifiers. Our proof that our protocol is zero-knowledge for *classical* verifiers only requires that the NP proof system used here is (likewise) zero-knowledge against classical verifiers; however, it becomes necessary to require post-quantum security of this proof system if we want our protocol also to be zero-knowledge for potentially quantum malicious verifiers.

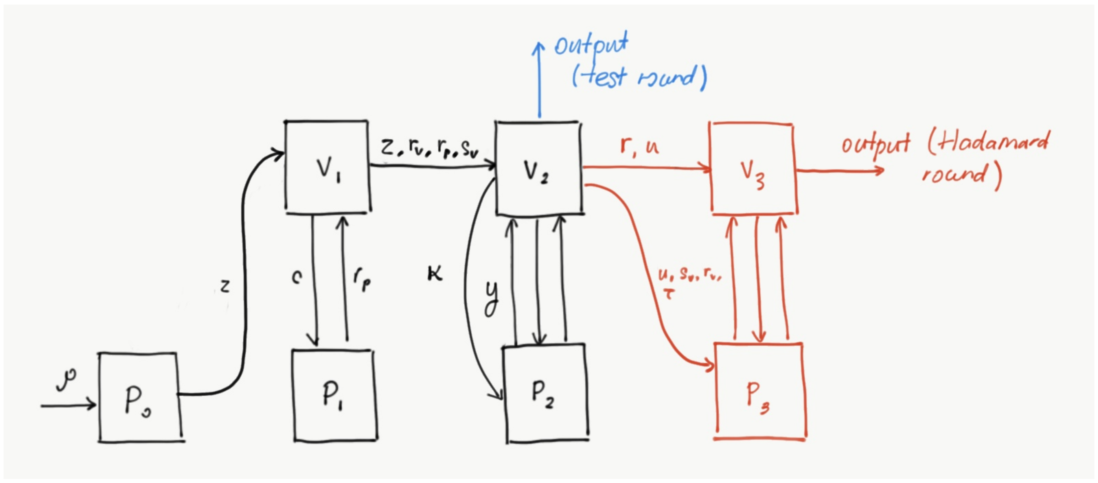


Figure 1: Diagrammatic representation of an honest execution of Protocol 3.3. We omit communication between the different parts of the prover for neatness, and we also omit the initial messages i (see Appendix C) from executions of the perfectly binding, quantum computationally concealing commitment protocol which we refer to in Assumptions 3.1. The blue parts of the diagram indicate what occurs in the case of a test round, and the red parts indicate what occurs in the case of a Hadamard round.

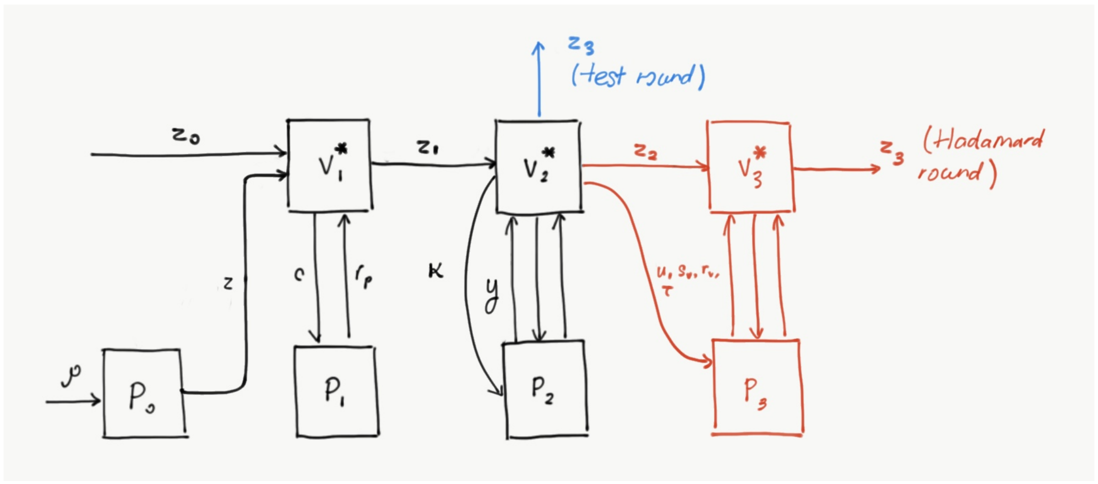


Figure 2: Diagrammatic representation of Protocol 3.3 with a cheating verifier. The cheating verifier V^* may take some (classical) auxiliary input Z_0, store auxiliary information (represented by Z_1 and Z_2), and produce a final output Z_3 that deviates from that specified by the protocol.

4 Soundness of protocol

Let the honest verifier of the argument system in Protocol 3.3 be denoted V, and let an arbitrary quantum polynomial-time prover with which V interacts be denoted $\mathbb{P}$. For this section, we will require notation from Section 5.3 of [Mah18], the proof of Theorem 8.6 of the same paper, and Section 4 of [BJSW16]. We will by and large introduce this notation as we proceed (and some of it has been introduced already in Sections 2.2 and 2.3, the sections containing outlines of the measurement protocol from [Mah18] and the zero-knowledge proof system from [BJSW16]), but the reader should refer to the above works if clarification is necessary.

We begin by making some preliminary definitions and proving a claim, from which the soundness of Protocol 3.3 (Lemma 4.6) will naturally follow. Firstly, we introduce some notation from Section 4 of [BJSW16]:

Definition 4.1 (Projection operators Π_0 and Π_1)**.** Define N as it is defined in Protocol 3.3. Let $\mathcal{D}_N^0$ be the set of all the classical codewords in the concatenated Steane code of Protocol 2.11 (or of Protocol 3.3) which are encodings of 0, and let $\mathcal{D}_N^1$ likewise be the set of classical codewords in the concatenated Steane code which encode 1. (See Definition 2.13, and Section A.6 of [BJSW16], for details about the concatenated Steane code. The first condition in Definition 2.13 will provide some motivation for the following definitions of Π_0 and Π_1.) Define

$$\Pi_0 = \sum_{x \in \mathcal{D}_N^0} |x\rangle\langle x| \ , \qquad \Pi_1 = \sum_{x \in \mathcal{D}_N^1} |x\rangle\langle x| \ .$$

Definition 4.2 (Projection operators Δ_0 and Δ_1)**.** Define N as it is defined in Protocol 3.3. Let Δ_0 and Δ_1 be the following projection operators:

$$\Delta_0 = \frac{I^{\otimes N} + Z^{\otimes N}}{2} \ , \qquad \Delta_1 = \frac{I^{\otimes N} - Z^{\otimes N}}{2} \ .$$

Δ_0 is the projection onto the space spanned by all even-parity computational basis states, and Δ_1 is its equivalent for odd-parity basis states. Note that, since all the codewords in $\mathcal{D}_0$ have even parity, and all the codewords in $\mathcal{D}_1$ have odd parity, it holds that $\Pi_0 \leq \Delta_0$ and that $\Pi_1 \leq \Delta_1$.

Definition 4.3 (The quantum channel Ξ)**.** Define a quantum channel mapping N qubits to one qubit as follows:

$$\Xi_N(\sigma) = \frac{\langle I^{\otimes N}, \sigma\rangle I + \langle X^{\otimes N}, \sigma\rangle X + \langle Y^{\otimes N}, \sigma\rangle Y + \langle Z^{\otimes N}, \sigma\rangle Z}{2} \ .$$

Loosely, Ξ_N can be thought of as a simplification of the decoding operator to the concatenated Steane code that the honest prover applies to its quantum witness in Protocol 2.11 (or in Protocol 3.3). Its adjoint is specified by

$$\Xi_N^*(\sigma) = \frac{\langle I, \sigma\rangle I^{\otimes N} + \langle X, \sigma\rangle X^{\otimes N} + \langle Y, \sigma\rangle Y^{\otimes N} + \langle Z, \sigma\rangle Z^{\otimes N}}{2} \ ,$$

and has the property that

$$\Xi_N^*(|0\rangle\langle 0|) = \Delta_0 \ , \qquad \Xi_N^*(|1\rangle\langle 1|) = \Delta_1 \ ,$$

a property which we will shortly use.

Let z be prover $\mathbb{P}$'s commitment string from step 2 of Protocol 3.3. Because the commitment protocol is perfectly binding, there exists a unique, well-defined tuple (π, a, b) and a string s_p such that $z = \mathsf{commit}((\pi, a, b), s_p)$.

Definition 4.4. For notational convenience, we define a quantum procedure M on a $2nN$-qubit state ρ as follows:

1. Apply $X^a Z^b$ to ρ, to obtain a state ρ'.
2. Apply π^{-1} to each $2N$-qubit block in the state ρ', to obtain a state ρ''.
3. Discard the last N qubits of each $2N$-qubit block in ρ'', to obtain a state ρ'''.
4. To each N-qubit block in ρ''', apply the map Ξ_N.

We also define the procedure $\tilde{M}$ as the application of the first four steps in M, again for notational convenience.

Intuitively, we think of M as an inverse to the prover's encoding procedure E. M may not actually invert the prover's encoding procedure, if the prover lied about the encoding key that it used when it sent the verifier $z = \mathsf{commit}((\pi, a, b), s_p)$; however, this is immaterial.

We now prove a claim from which the soundness of Protocol 3.3 will follow. Before we do so, however, we make a remark about notation for clarity. When we write 'V accepts the distribution $D_{\xi,h}$ with probability p' (or similar phrases), we mean that, in [Mah18]'s notation from section 8.2,

$$\sum_{h \in \{0,1\}^{2nN}} v_h(1 - \tilde{p}_h(D_{\xi,h})) = p.$$

Here, h represents the verifier's choice of measurement bases, as before; v_h is the probability that the honest verifier will select the basis choice h, and $1 - \tilde{p}_h(D)$ is defined, for any distribution D over measurement outcomes $m \in \{0,1\}^{2nN}$, as the probability that the honest verifier will accept a string drawn from D on basis choice h. (When we refer to the latter probability, we assume, following [BJSW16, Section 4], that the prover behaves optimally—in terms of maximising the verifier's eventual probability of acceptance—after the verifier sends it measurement outcomes at the end of step 6 in Protocol 3.3. For the purposes of the present soundness analysis, therefore, we can imagine that the verifier checks the predicate Q itself after step 6, instead of relying on the prover to prove to it during step 8 that Q is satisfied.)

Claim 4.5. *Suppose there exists a quantum state ξ such that the honest verifier V accepts the distribution $D_{\xi,h}$ with probability p. Then the state $M(\xi)$ is accepted by the verifier of Protocol 2.4 with probability at least p.*

Proof. Fix a choice of r (see step 3 of Protocol 3.3 for a definition of r). Let $\mathcal{Z}_r$ be the subset of $\{0,1\}^n$ such that the verifier of Protocol 2.4 accepts if and only if the n-bit string that results from concatenating the measurement results it obtains in step 4 of said protocol is a member of $\mathcal{Z}_r$. It is unimportant to the analysis what $\mathcal{Z}_r$ actually is; it matters only that it is well-defined.

For this choice of r, we can express the probability that the verifier of Protocol 2.4 accepts a state τ as

$$\sum_{z \in \mathcal{Z}_r} \Big\langle U_r^* \left|z_1, \ldots, z_n\right\rangle \left\langle z_1, \ldots, z_n\right| U_r, \tau \Big\rangle.$$

(Though only $2m$ of the n qubits in τ are relevant to U_r, we assume here for notational simplicity that U_r is a gate on n qubits, and that the verifier measures all n qubits of $U_r \tau$ and ignores those measurement results which are irrelevant.)

For the same choice of r, we can express the probability that the verifier V from Protocol 3.3 will eventually accept the distribution $D_{\xi,h}$ as

$$p_r = \sum_{z \in \mathcal{Z}_r} \Big\langle (U_r^*)^{\otimes N} (\Pi_{z_1} \otimes \cdots \otimes \Pi_{z_n}) (U_r)^{\otimes N}, \tilde{M}(\xi) \Big\rangle .$$

Following [BJSW16], we note that

$$\begin{aligned}
&\sum_{z\in\mathcal{Z}_r}\Big\langle (U_r^*)^{\otimes N}(\Pi_{z_1}\otimes\cdots\otimes\Pi_{z_n})(U_r)^{\otimes N}, \tilde{M}(\xi)\Big\rangle\\
&\le \sum_{z\in\mathcal{Z}_r}\Big\langle (U_r^*)^{\otimes N}(\Delta_{z_1}\otimes\cdots\otimes\Delta_{z_n})(U_r)^{\otimes N}, \tilde{M}(\xi)\Big\rangle\\
&= \sum_{z\in\mathcal{Z}_r}\Big\langle (U_r^*)^{\otimes N}\Big(\Xi_N^*(|z_1\rangle\langle z_1|)\otimes\cdots\otimes\Xi_N^*(|z_n\rangle\langle z_n|)\Big)(U_r)^{\otimes N}, \tilde{M}(\xi)\Big\rangle\\
&= \sum_{z\in\mathcal{Z}_r}\Big\langle (\Xi_N^{\otimes n})^*(U_r^*)^{\otimes N}|z_1,\ldots,z_n\rangle\langle z_1,\ldots,z_n|(U_r)^{\otimes N}, \tilde{M}(\xi)\Big\rangle\\
&= \sum_{z\in\mathcal{Z}_r}\Big\langle U_r^*|z_1,\ldots,z_n\rangle\langle z_1,\ldots,z_n|U_r, M(\xi)\Big\rangle\ .
\end{aligned}$$

We conclude that, if the distribution $D_{\xi,h}$ is accepted by V with probability $p=\sum_r v_r p_r=\sum_h v_h(1-\tilde{p}_h(D_{\xi,h}))$ (where v_r is the probability that a given r will be chosen, and the second expression is simply a formulation in alternative notation of the first), the state $M(\xi)$ is accepted by the verifier of Protocol 2.4 with probability at least p. □

Now we turn to arguing that Protocol 3.3 has a soundness parameter s which is negligibly close to $\frac{3}{4}$.

Lemma 4.6. *Suppose that the instance $x=(H,a,b)$ of the 2-local XZ Hamiltonian problem that is provided as input to the verifier and prover in Protocol 3.3 is a no-instance, i.e. the ground energy of H is larger than b. Then, provided that Assumptions 3.1 hold, the probability that the honest verifier V accepts in Protocol 3.3 after an interaction with any quantum polynomial-time prover $\mathbb{P}$ is at most $\frac{3}{4}+\mathsf{negl}(|x|)$.*

Proof. Claim 7.1 of [Mah18] guarantees that, for any arbitrary quantum polynomial-time prover $\mathbb{P}$ who executes the measurement protocol with V, there exists a state ξ, a prover $\mathbb{P}'$ and a negligible function μ such that

$$\begin{aligned}\|D^C_{\mathbb{P},h}-D_{\mathbb{P}',h}\|_{TV}\le\sqrt{p_{h,T}}+p_{h,H}+\mu\ ,\quad\text{and}\\ D_{\mathbb{P}',h}\approx_c D_{\xi,h}\ .\end{aligned}\tag{1}$$

(See the paragraph immediately above Claim 2.10 for relevant notation.)

It follows from (1) that, if V accepts the distribution $D_{\mathbb{P}',h}$ with probability p, it must accept the distribution $D_{\xi,h}$ with probability $p-\mathsf{negl}(N)$, because the two are computationally indistinguishable and the verifier V is efficient. Therefore (using Claim 4.5), if V accepts $D_{\mathbb{P}',h}$ with probability p, the verifier of Protocol 8.3 from [Mah18] accepts the state $M(\xi)$ with probability at least $p-\mathsf{negl}(N)$. By the soundness of Protocol 2.4 (Claim 2.5), we conclude that $p=\mathsf{negl}(N)$ when the problem Hamiltonian is a no-instance.

We now apply a similar argument to that which is used in Section 8.2 of [Mah18] in order to establish an upper bound on the probability ϕ that V accepts $\mathbb{P}$ in a no-instance. Let $E^H_{\mathbb{P},h}$ denote the event that the verifier V does not reject the prover labelled $\mathbb{P}$ in a Hadamard round indexed by h *during the measurement protocol phase* of Protocol 3.3. Let $E^T_{\mathbb{P},h}$ denote the analogous event in a test round. Furthermore, let $E_{\mathbb{P},h}$ denote the event that the verifier accepts the prover $\mathbb{P}$ in the last step of Protocol 3.3. The total probability that V accepts $\mathbb{P}$ is the average, over all possible basis choices h, of the probability that V accepts $\mathbb{P}$ after a test round indexed by h, plus the probability that V accepts $\mathbb{P}$ after a Hadamard round indexed by h. As such,

$$\begin{aligned}\phi &= \sum_{h\in\{0,1\}^{2nN}} v_h\big(\,\frac{1}{2}Pr[E^T_{\mathbb{P},h}] + \frac{1}{2}Pr[E^H_{\mathbb{P},h}\cap E_{\mathbb{P},h}]\,\big)\\ &= \sum_{h\in\{0,1\}^{2nN}} v_h\big(\,\frac{1}{2}Pr[E^T_{\mathbb{P},h}] + \frac{1}{2}Pr[E^H_{\mathbb{P},h}]Pr[E_{\mathbb{P},h}|E^H_{\mathbb{P},h}]\,\big)\\ &= \sum_{h\in\{0,1\}^{2nN}} v_h\big(\,\frac{1}{2}(1-p_{h,T}) + \frac{1}{2}(1-p_{h,H})(1-\tilde{p}_h(D^C_{\mathbb{P},h}))\,\big)\,.\end{aligned}$$

Since Lemma 3.1 and Claim 7.1 of [Mah18] taken together yield the inequality

$$\tilde{p}_h(D_{\mathbb{P}',h}) - \tilde{p}_h(D^C_{\mathbb{P},h}) \le \|D^C_{\mathbb{P},h} - D_{\mathbb{P},h}\|_{TV} \le \sqrt{p_{h,T}} + p_{h,H} + \mu\,,$$

it follows that

$$\begin{aligned}\phi &\le \sum_{h\in\{0,1\}^{2nN}} v_h\big(\,\frac{1}{2}(1-p_{h,T}) + \frac{1}{2}(1-p_{h,H})(1-\tilde{p}_h(D_{\mathbb{P}',h}) + \sqrt{p_{h,T}} + p_{h,H} + \mu)\,\big)\\ &\le \frac{1}{2}\mu + \frac{1}{2}\sum_{h\in\{0,1\}^{2nN}} v_h(1-p_{h,T} + (1-p_{h,H})(p_{h,H} + \sqrt{p_{h,T}})) + \frac{1}{2}\sum_{h\in\{0,1\}^{2nN}} v_h(1-\tilde{p}_h(D_{\mathbb{P}',h}))\\ &\le \frac{1}{2}\mu + \frac{3}{4} + \frac{1}{2}p\,.\end{aligned}$$

We conclude that Protocol 3.3 has a soundness parameter s which is negligibly close to $\frac{3}{4}$. □

5 Zero-knowledge property of protocol

In this section, we establish that Protocol 3.3 is zero-knowledge against arbitrary classical probabilistic polynomial time (CPPT) verifiers. Specifically, we show the following:

Lemma 5.1. *Suppose that the instance $x = (H, a, b)$ of the 2-local XZ Hamiltonian problem that is provided as input to the verifier and prover in Protocol 3.3 is a yes-instance, i.e. the ground energy of H is smaller than a. Then (provided that Assumptions 3.1 hold), for any arbitrary CPPT verifier V^*, there exists a simulator S which can be implemented in CPPT such that the distribution of V^*'s final output after its interaction with the honest prover P in Protocol 3.3 is (classical) computationally indistinguishable from S's output distribution.*

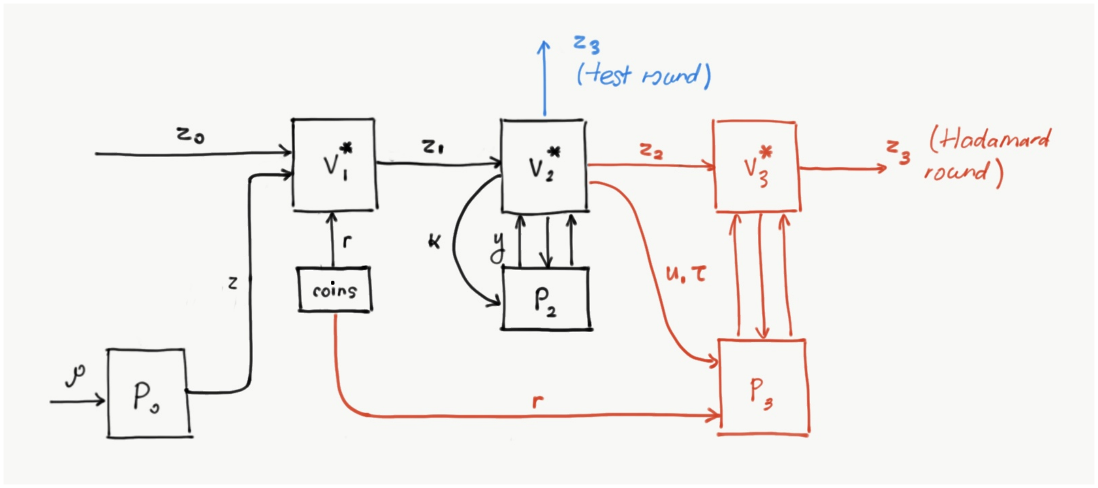
Remark 5.2. Lemma 5.1 formulates the zero-knowledge property in terms of classical verifiers and computational indistinguishability against classical distinguishers, because this is the most natural setting for a protocol in which verifier and interaction are classical. However, the same proof can be adapted to show that, for any quantum polynomial-time verifier executing Protocol 3.3, there exists a quantum polynomial-time simulator whose output is QPT indistinguishable in yes-instances from that of the verifier. (In particular, the latter follows from the fact that the second item in Assumptions 3.1 implies an NP proof system which is zero-knowledge against quantum polynomial-time verifiers, an implication shown to be true in [Wat09].)

We show that Protocol 3.3 is zero-knowledge by replacing the components of the honest prover with components of a simulator one at a time, and demonstrating that, when the input is a yes-instance, the dishonest verifier's output after each replacement is made is at the least computationally indistinguishable from its output before. The argument proceeds in two stages. In the first, we show that the honest prover can be replaced by a *quantum polynomial-time* simulator that does not have access to the witness ρ. In the second, we de-quantise the simulator to show that the entire execution can be simulated by a classical simulator who likewise does not have access to ρ. (The latter is desirable because the verifier is a classical entity.)

We begin with the protocol execution between the honest prover P and an arbitrary cheating verifier V^*, the latter of whom may take some (classical) auxiliary input Z_0, store information (represented by Z_1 and Z_2), and produce an arbitrary final output Z_3. A diagram representing the interaction between V^* and P can be found in Figure **??**.

5.1 Eliminating the coin-flipping protocol

Our first step in constructing a simulator is to eliminate the coin-flipping protocol, which is designed to produce a trusted random string r, and replace it with the generation of a truly random string. (This step is entirely analogous to step 1 of Section 5 in [BJSW16], and we omit the analysis.) The new diagram is shown below. In this diagram, coins represents a trusted procedure that samples a uniformly random string r of the appropriate length.



5.2 Introducing an intermediary

Our next step is to introduce an *intermediary*, denoted by I, which pretends—to the cheating verifier of Protocol 3.3—to be its prover P, while simultaneously playing the role of *verifier* to the prover from the zero-knowledge proof system of Protocol 2.15 [9]. (We denote the honest prover and honest verifier for the proof system of Protocol 2.15 by $\mathcal{P}$ and $\mathcal{V}$, respectively, to distinguish them from the prover(s) P and verifier(s) V of the classical-verifier protocol currently under consideration.) We remark, for clarity, that I is a quantum polynomial-time procedure. The essential idea of this section is that I will behave so it is impossible for the classical verifier V to tell whether it is interacting with the intermediary or with its honest prover. (We achieve this simply by making I output exactly the same things that P would.) Given that this is so, the map that V implements from its input to its output, including its auxiliary registers, cannot possibly be different in the previous section as compared to this section.

[9]Protocol 2.15 is identical in structure to the protocol presented in [BJSW16]. We refer the reader to Figure 4 in that paper for a diagram representing the appropriate interactions.

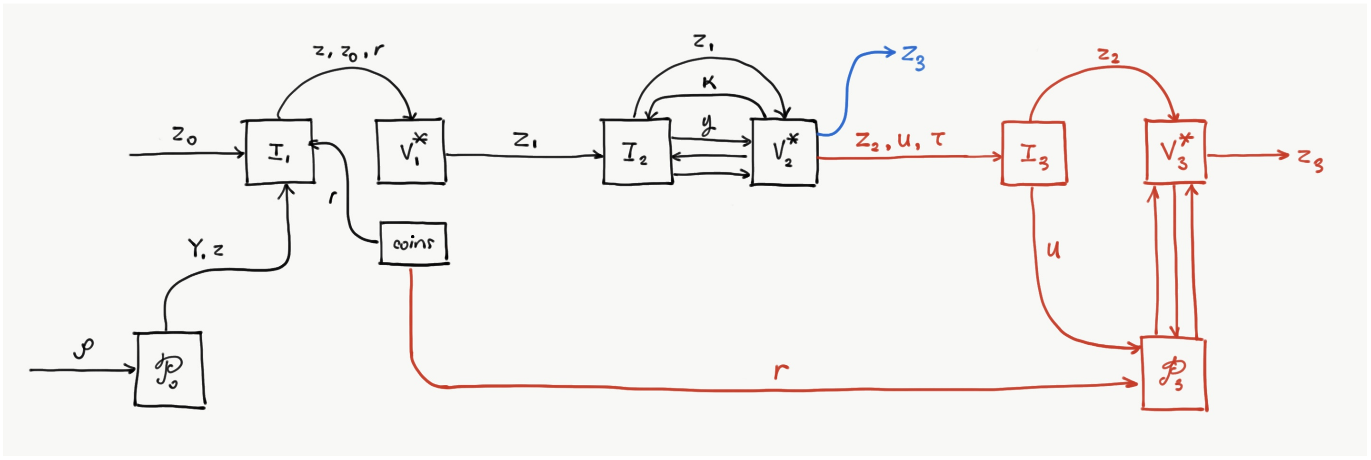


Figure 3: The intermediary interacting with the honest prover from the proof system of Protocol 2.15, denoted by $\mathcal{P}$, and also with the cheating classical verifier V^*. I_1 receives the encoded quantum witness, which we have denoted by Y, from $\mathcal{P}$, in addition to $\mathcal{P}$'s commitment z. It then sends z to V_1^*, along with Z_1, the auxiliary input that V_1^* is supposed to receive, and r, the random string generated by coins. I_2 passes on any output V_1^* produces to V_2^*, performs itself the procedure for committing to a quantum state from [Mah18], and executes the measurement protocol with V_2^*. I_3 receives the measurement outcomes u and the trapdoors τ from V_2^*, and checks whether the trapdoors are valid. If they are invalid, it aborts directly; if they are valid, it sends u on to $\mathcal{P}_3$ and passes Z_2 to V_3^*, so that $\mathcal{P}_3$ and V_3^* can execute the NP zero-knowledge proof protocol. (Each part of I should also send everything it knows to its successor, but we have omitted these communications for the sake of cleanliness, as we omitted the communication between parts of the prover in previous diagrams.)

5.3 Simulating the protocol with a quantum simulator

We now note that Figure 3 looks exactly like Figure 4 from [BJSW16], if we consider the intermediary I and the cheating classical verifier V^* taken together to be a cheating verifier $\mathcal{V}'$ for the proof system of Protocol 2.15.

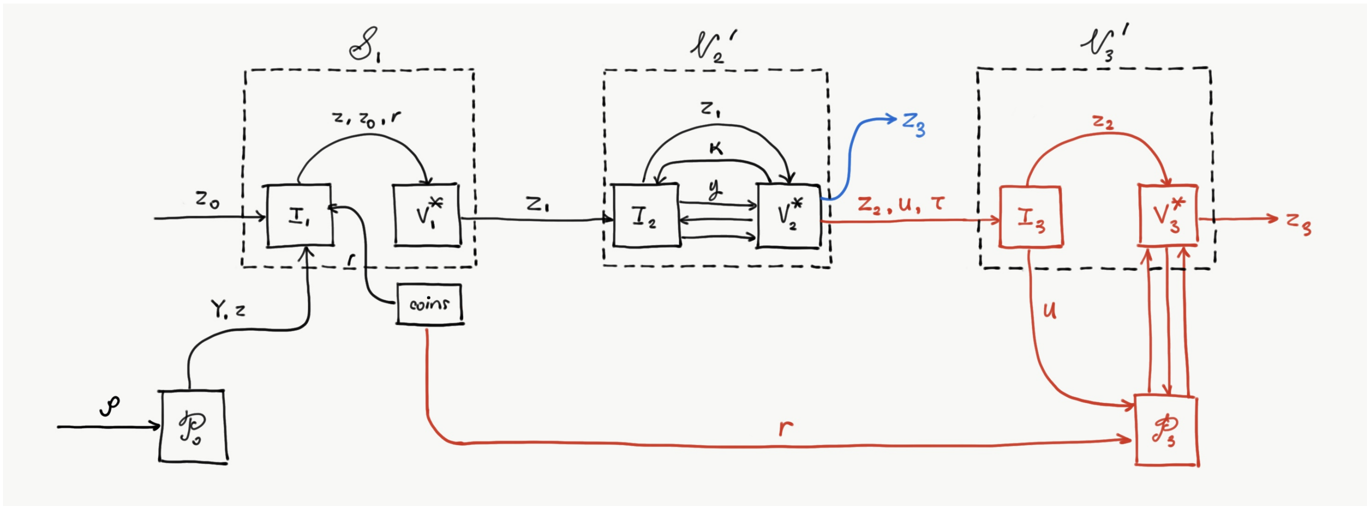


Figure 4: Compare to Figure 4 of [BJSW16]. Note that $\mathcal{S}_1$ includes the behaviour of an arbitrary $\mathcal{V}_1'$; the reason it is called $\mathcal{S}_1$ and not $\mathcal{V}_1'$ is because $\mathcal{V}_1'$ obtains r from a coin-flipping protocol, while $\mathcal{S}_1$ generates r using coins. In all other respects, $\mathcal{S}_1$ is the same as $\mathcal{V}_1'$.

Using similar reasoning as in [BJSW16] (and recalling that, by Lemma 2.18, it still works when the Hamiltonian being verified is an XZ Hamiltonian), therefore, we conclude that we can replace ρ in Figure 4 with ρ_r—where ρ_r is a quantum state specifically designed to pass the challenge indexed by r—without affecting the verifier's output distribution (to within computational indistinguishability). See Remark 2.6 for a procedure that explicitly constructs ρ_r. Note that, if our objective was to achieve a *quantum* simulation without knowing the witness state ρ, our task would already be finished at this step. However, our verifier is classical; therefore, in order to prove that our classical verifier's interaction with its prover does not impart to it any knowledge (apart from the fact that the problem instance is a yes-instance) that it could not have generated itself, we need to achieve a *classical* simulation of the argument system.

5.4 Simulating the protocol with a classical simulator

5.4.1 Replacing $\mathcal{P}_0$ and I_1

If we want to simulate the situation in Figure 4 classically, then we need to de-quantise $\mathcal{P}_0$, I_1 and I_2. (I_3 and $\mathcal{P}_3$ are already classical.) Our first step is to replace $\mathcal{P}_0$ and I_1 with a single classical entity, I_1'.

I_1' simply chooses encoding keys (t, π, a, b) and generates z, a commitment to the encoding keys (π, a, b). It then sends z, r and Z_1 to V_1^*, as I_1 would have. Because I_1' has exactly the same output as I_1, the verifier's output in Figure 5 is the same as its output in Figure 4. (We assume that the still-quantum I_2 now generates ρ_r for itself.)

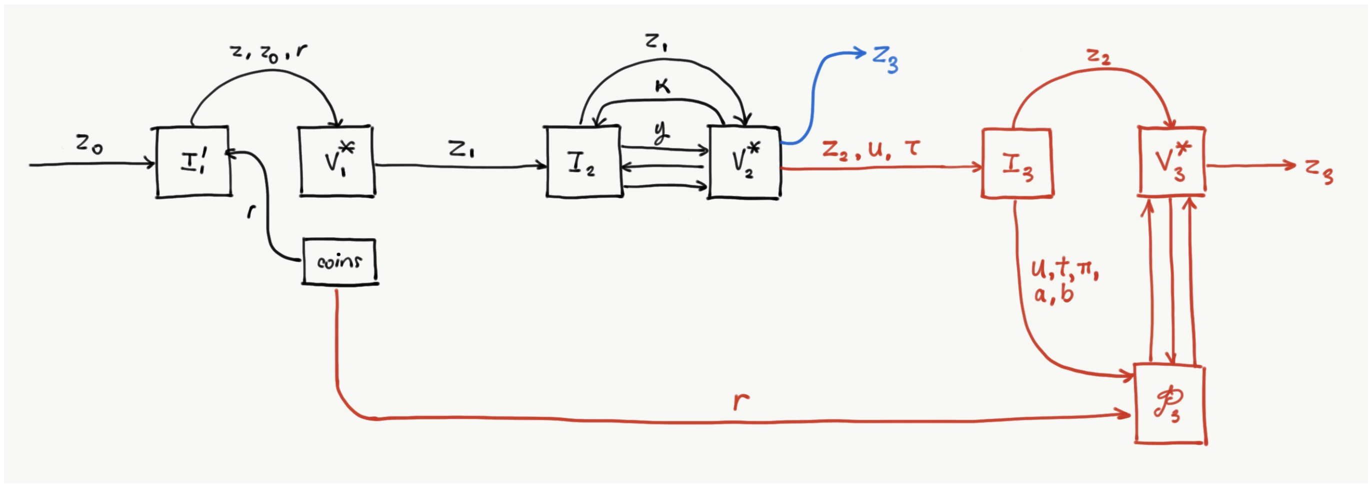


Figure 5: $\mathcal{P}_0$ and I_1 have been replaced by I_1'.

5.4.2 Some simplifications (which make it possible to de-quantise I_2)

Following [BJSW16], we make some alterations to Figure 5 that will allow us to eventually de-quantise I_2. The alterations are as follows:

1. Replace V_3^* and $\mathcal{P}_3$ with an efficient simulation S_3. (An efficient simulation of the NP proof protocol execution between V_3^* and $\mathcal{P}_3$ is guaranteed to exist because the NP proof protocol is zero-knowledge.) Recall that the statement $\mathcal{P}_3$ is meant to prove to V_3^* in a zero-knowledge way is as follows: 'There exists a string s_p and an encoding key (t, π, a, b) such that $z = \mathsf{commit}((\pi, a, b), s_p)$ and $Q(t, \pi, a, b, r, u) = 1$.' The zero-knowledge property of the NP proof system guarantees that, for yes-instances, the output of S_3 is indistinguishable from the output of the protocol execution between V_3^* and P_3. In our case, I_1' always holds s_p and (π, a, b) such that $z = \mathsf{commit}((\pi, a, b), s_p)$, and the honest prover will abort the protocol if $Q(t, \pi, a, b, r, u) = 0$. Therefore, whenever the prover does not abort, the output of S_3 is computationally indistinguishable from that of V_3^* and P_3. We assume, following [BJSW16], that S_3 also behaves as V_3^* would when the prover aborts. If it does, then Figure 6 is computationally indistinguishable from Figure 5.

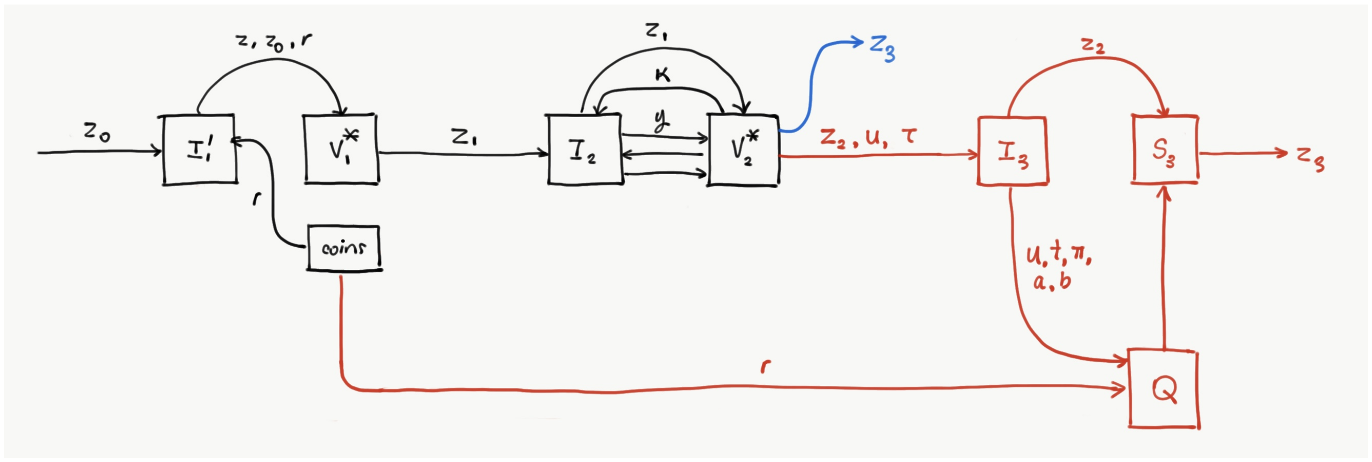


Figure 6: V_3^* and $\mathcal{P}_3$ have been replaced by S_3. Note that S_3 does not require access to the witness (s_p, t, π, a, b), and so s_p can be discarded immediately after I_1' is run.

2. Replace the generation of the genuine commitment z with the generation of a commitment $z' = \mathsf{commit}((\pi_0, a_0, b_0), s_p)$, where π_0, a_0 and b_0 are fixed strings independent of the encoding key (t, π, a, b) that I_1' chooses. Because the commitment protocol is (computationally) concealing, and the commitment is never opened (recall that s_p is discarded after I_1' is run), V_1^* should not be able to tell (computationally speaking) that z has been replaced by z'.

   The genuine encoding key is still used to evaluate the predicate Q. Note that, because z has been replaced with z', the statement for which S_3 must simulate the execution of a zero-knowledge proof between V_3^* and $\mathcal{P}_3$ is now as follows: 'There exists a string s_p and an encoding key (t, π, a, b) such that $z' = \mathsf{commit}((\pi, a, b), s_p)$ and $Q(t, \pi, a, b, r, u) = 1$.' This statement is, in general, no longer true, because the commitment protocol is perfectly binding. However, if the predicate Q is still satisfied for the encoding key (t, π, a, b) that I_3 sent, then S_3 will proceed to generate a transcript for the no-instance that is computationally indistinguishable from a transcript for a yes-instance. If Q is no longer satisfied, then S_3 will abort, as before. In effect, therefore, the cheating verifier V^* will not be able to tell (up to computational indistinguishability) that z has been replaced by z', and that the NP statement being 'proven' to it is no longer true.

5.4.3 De-quantising I_2

We now replace I_2 with a classical entity I_2'. In the process, we require modifications to the behaviour of I_3.

Knowing r, I_2' can calculate for itself what ρ_r should be, though it cannot physically produce this state. As we noted in Remark 2.6, ρ_r is a simple state: it is merely the tensor product of $|0\rangle, |1\rangle, |+\rangle$ and $|-\rangle$ qubits. Applying the concatenated Steane code to ρ_r will then result in a tensor product of N-qubit states that look like

$$\sum_{x \in \mathcal{D}_N^0} |x\rangle\,, \quad \sum_{x \in \mathcal{D}_N^1} |x\rangle\,, \quad \sum_{x \in \mathcal{D}_N^0} |x\rangle + \sum_{x \in \mathcal{D}_N^1} |x\rangle\,, \quad \text{and} \quad \sum_{x \in \mathcal{D}_N^0} |x\rangle - \sum_{x \in \mathcal{D}_N^1} |x\rangle\,, \tag{2}$$

after appropriate normalisation.

A brief argument will suffice to establish that it is possible to classically simulate standard or Hadamard basis measurements on the qubits in $E(\rho_r)$. Each qubit of $E(\rho_r)$ is either an encoding qubit or a trap qubit, up to the application of a random single-qubit Pauli operator. Simulating standard-basis measurements of encoding qubits is classically feasible, because $\mathcal{D}_N^0$ and $\mathcal{D}_N^1$ are polynomially sized, and the expressions in (2) only involve superpositions over those sets with equal-magnitude coefficients. Simulating standard-basis measurements of trap qubits, which are always initialised either to $|0\rangle$ or $|+\rangle$, is trivially feasible.

To simulate a Hadamard basis measurement, we can take advantage of the transversal properties of the encoding scheme, and apply H before we apply the concatenated Steane code. Denote the application of the concatenated Steane code to ρ_r by $S(\rho_r)$. We have that

$$S(H^{\otimes n}\rho_r H^{\otimes n}) = H^{\otimes nN} S(\rho_r) H^{\otimes nN}$$

by transversality. To simulate a Hadamard basis measurement of $E(\rho_r)$, we then

1. Apply $H^{\otimes n}$ to ρ_r. This is easy to classically simulate, because ρ_r is a tensor product of $|0\rangle, |1\rangle, |+\rangle$ and $|-\rangle$ qubits.
2. Apply the concatenated Steane code to $H^{\otimes n}\rho_r H^{\otimes n}$. Simulating this is classically feasible, by the same argument that we used for standard basis measurements, because $H^{\otimes n}\rho_r H^{\otimes n}$ is still a tensor product of $|0\rangle, |1\rangle, |+\rangle$ and $|-\rangle$ qubits.
3. Concatenate trap qubits to each N-qubit block in $S(H^{\otimes n}\rho_r H^{\otimes n}) = H^{\otimes nN} S(\rho_r) H^{\otimes nN}$. Simulate the application of H to each trap qubit (which is, once again, classically easy to do because each trap qubit is initialised either to $|0\rangle$ or to $|+\rangle$).
4. Apply the permutation π to each $2N$-tuple.
5. Simulate a standard basis measurement of the result.
6. XOR the string b to the measurement outcome (b was previously the Z-key for the Pauli one-time pad).

Having established that it is possible to classically simulate standard and Hadamard basis measurements of the qubits in $E(\rho_r)$, we now describe the procedure that the classical I_2' should follow for each qubit i in the state $E(\rho_r)$.

1. During the commitment phase, I_2' simulates a standard basis measurement on the ith qubit, obtains a simulated measurement result β_i, and then chooses a uniformly random preimage x_i from the domain of the function specified by κ_i. It applies the function specified by κ_i to $\beta_i \| x_i$ and sets $y_i = \eta_{\kappa_i}(\beta_i \| x_i)$.
2. If the verifier requests a test round, I_2' sends $\beta_i \| x_i$ to the verifier. This is exactly what the quantum prover I_2 would send in the case of a test round, so the verifier cannot tell that it is interacting with I_2' instead of I_2.
3. If the verifier requests a Hadamard round, I_2' sends a uniformly random string $s_i \in \{0,1\}^{w+1}$ to the verifier, where w is the length of the preimages. In the same situation, the quantum I_2 would have sent Hadamard basis measurements of the $w+1$ qubits in the ith committed qubit in $E(\rho)$ and its associated preimage register.
   (a) If $h_i = 0$, the outcomes of these measurements are uniformly distributed and thus indistinguishable from the distribution of strings s_i reported by I_2'.
   (b) Let $|\psi_i\rangle$ be the state of the ith qubit of $E(\rho)$, let $x_{0,i}$ and $x_{1,i}$ be the two preimages to y_i under the function f_{κ_i}, and let b_i be the ith bit of the one-time-pad Z-key b from I_1''s encoding key (t, π, a, b). If $h_i = 1$, the outcomes of I_2's Hadamard basis measurements can be represented as a tuple (β_i, d_i), where d_i is uniformly random, and

$$\beta_i = d_i \cdot (x_{0,i} \oplus x_{1,i}) \oplus b_i \oplus \mathsf{Meas}(H|\psi_i\rangle) .$$

   (Meas here denotes a standard basis measurement.)

Note that the distribution over (b_i, β_i, d_i) which one would obtain by measuring $|\psi_i\rangle$ in the Hadamard basis, choosing d_i and b_i uniformly at random, and letting

$$\beta_i = d_i \cdot (x_{0,i} \oplus x_{1,i}) \oplus b_i \oplus \mathsf{Meas}(H\,|\psi_i\rangle)$$

is equivalent to the one that one would obtain choosing a uniformly random s_i, measuring $|\psi_i\rangle$ in the Hadamard basis, calculating

$$b_i = s_{i,1} \oplus d_i \cdot (x_{0,i} \oplus x_{1,i}) \oplus \mathsf{Meas}(H\,|\psi_i\rangle)\ ,$$

and finally setting $\beta_i = s_{i,1},\ d_i = s_{i,2} \cdots s_{i,w+1}$.

The former set of actions is equivalent to the set of actions that I_2 performs. The latter set of actions is (as we will shortly show) classically feasible provided that we have the verifier's trapdoors. Note that I_2' only needs to send the verifier s_i, and can rely on its successor I_3, who *will* have access to the verifier's trapdoors, to calculate the bits b_i retroactively. It follows that, given that I_3 can produce correct bits b_i (we will shortly show that it can), the distribution of strings reported by I' is identical to the distribution of outcomes reported by I.

Having established that I_2' and I_2 are the same from V_2^*'s perspective (meaning that it must have the same behaviour that it did in Figure 5 after I_2 is replaced with I_2'), it remains to ensure that the choice of the one-time pad Z-key b is consistent with the s_i that I_2' picked. We relegate the task of making this choice to I_3', our new version of I_3, because it has access to the verifier's trapdoors τ. If any of the trapdoors that it receives from the verifier are invalid, I_3' aborts, as specified in Protocol 3.3. ('Validity', here, means what we defined it to mean in step 8 of Protocol 3.3: that, for each y_i, the trapdoor τ_{κ_i} returns all the preimages to y_i under the function specified by κ_i, and no others. See Definition 2.9 for a definition of the phrase 'preimages to y'.) Presuming upon valid trapdoors, I_3' then deduces the verifier's choices of measurement basis, h, from τ. Given that the trapdoors are valid, I_3' can be confident that its deductions in this regard will lead it to behave in the same way that the honest prover would: because the trapdoors return all the preimages to y_i under κ_i and no others, I_3' knows exactly which superpositions the honest prover would have obtained during the measurement protocol after following the verifier's instructions.

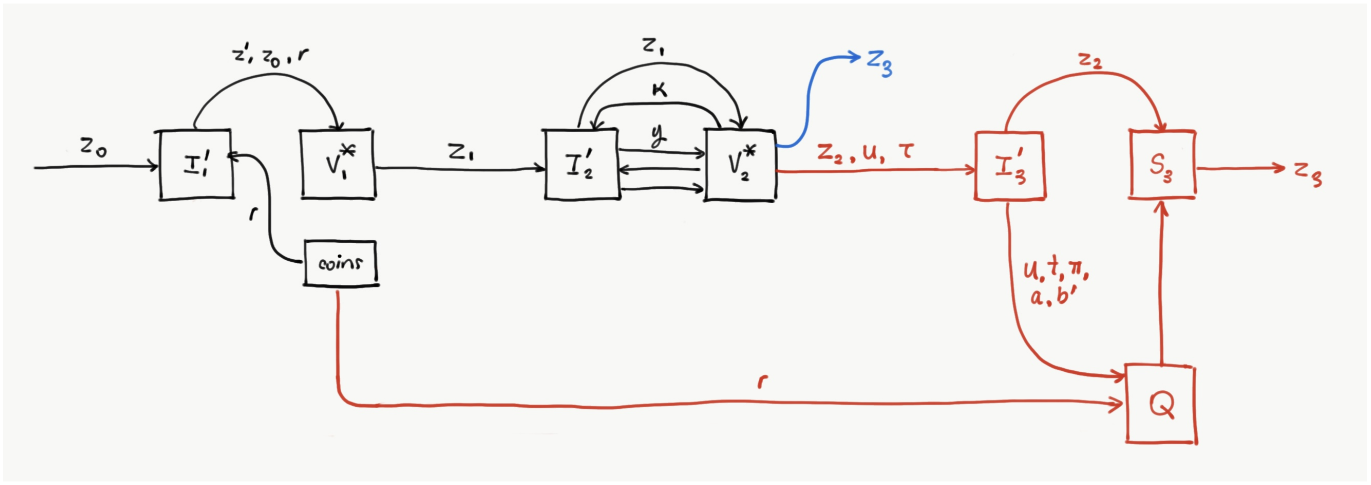
I_3' subsequently executes the following procedure for all i such that $h_i = 1$:

1. Set d_i to be the last w bits of s_i, and compute $d_i \cdot (x_{0,i} \oplus x_{1,i})$ using the trapdoor τ_{κ_i}.
2. Simulate a standard basis measurement of $HX^{a_i}\,|\psi_i^*\rangle$. Denote the result by β_i. (Here, a_i refers to the ith bit of a, where a is taken from I_1''s initial choice of one-time pad keys. $|\psi_i^*\rangle$, meanwhile, denotes the ith qubit of the state obtained by applying the first three steps of E—but not the last—to ρ_r.)
3. Set b_i' (the ith bit of b', the new Z-key for the one-time pad) to be equal to $\beta_i \oplus s_{i,1} \oplus d_i \cdot (x_{0,i} \oplus x_{1,i})$ (where $s_{i,1}$ refers to the first bit of s_i). This will cause the equation $\mathsf{Meas}(H\,|\psi_i\rangle) \oplus d_i \cdot (x_{0,i} \oplus x_{1,i}) = s_{i,1}$ to be satisfied:

$$\begin{aligned}
&\mathsf{Meas}(H\,|\psi_i\rangle) \oplus d_i \cdot (x_{0,i} \oplus x_{1,i}) = s_{i,1} \\
\iff\ &\mathsf{Meas}(HZ^{b_i}X^{a_i}\,|\psi_i^*\rangle) \oplus d_i \cdot (x_{0,i} \oplus x_{1,i}) = s_{i,1} \\
\iff\ &b_i \oplus \mathsf{Meas}(HX^{a_i}\,|\psi_i^*\rangle) \oplus d_i \cdot (x_{0,i} \oplus x_{1,i}) = s_{i,1} \\
\iff\ &b_i \oplus \beta_i \oplus d_i \cdot (x_{0,i} \oplus x_{1,i}) = s_{i,1} \\
\iff\ &b_i = \beta_i \oplus s_{i,1} \oplus d_i \cdot (x_{0,i} \oplus x_{1,i}).
\end{aligned}$$

Having done this, I_3' then feeds (t, π, a, b') into Q. In all other respects I_3' behaves the same way that I_3 did.

The final simulation will be as follows:



Since all the entities in this simulation are classical and efficient, and none have access to information about the witness state ρ, it follows that the protocol is zero-knowledge.

A The trapdoor check can be implemented efficiently

For this proof sketch, we rely on the specific properties of the LWE-based ETCFF family that is used in [Mah18]. As such, we begin by briefly introducing at a high level how the instantiations of the keys κ and the trapdoors τ for noisy trapdoor claw-free (f) and trapdoor injective (g) functions, whose properties we have relied upon in a black-box way for the rest of this work, are respectively achieved. For details, we refer the reader to Section 9 of [Mah18].

The key (κ_1, κ_2) for an f (two-to-one) function is $(A, As + e)$, where A is an LWE matrix and e is an error vector with small and bounded magnitude. The key for a g (injective) function is (A, u), where u is a random vector not of the form $As + e$ for any e of small enough magnitude. (The distribution of u is uniform over all vectors that satisfy this latter requirement.) The trapdoor in both cases is the trapdoor for A proposed by [MP11]. Theorem 5.1 of [MP11] states that there exists a pair of efficient algorithms $\mathsf{GenTrap}$ and Invert which, respectively, generates a matrix A along with its trapdoor τ_A, such that A is computationally indistinguishable from a uniformly random matrix of the correct dimensions; and, given $As+e$ (with e of sufficiently small magnitude) and the trapdoor τ_A for A, returns (s, e) with high probability.

The functions f_κ and g_κ both take as input a bit b and a vector x and output a probability distribution. Given a sample y from one such probability distribution Y, the trapdoor τ_A can be used to recover the tuple(s) (b, x) which map to Y under the function specified by κ. The functions f_κ and g_κ can be defined (loosely, but the most intuitively) as follows:

Definition A.1 (Informal definition of trapdoor claw-free and trapdoor injective functions)**.**

(a) $f_\kappa(b, x) = Ax + e_0 + b \cdot (As + e)$,

where e_0 is distributed as a Gaussian with small Gaussian parameter

(b) $g_\kappa(b, x) = Ax + e_0 + b \cdot u$.

What the above notation means, in a slightly more precise sense, is that one samples from the distribution determined by the input (b, x) and the function key $\kappa = (\kappa_1, \kappa_2)$ by sampling e_0 from the appropriate Gaussian and then computing $\kappa_1 x + e_0 + b \cdot (\kappa_2)$. It follows from hardness of the (decisional) LWE assumption that the keys for the f functions and the keys for the g functions are computationally indistinguishable.

Now we state, and prove, a claim that bears upon the one we eventually want to make.

Claim A.2. *Let A be an LWE matrix, let $\kappa = (A, \kappa_2)$, and let the function η_κ be defined by $\eta_\kappa(b, x) = Ax + e_0 + b \cdot \kappa_2$. (The output of η_κ is, as in Definition A.1, a probability distribution.) Suppose that the trapdoor τ_A inverts the matrix A, in the sense that, given $r = As + e$ for some e of sufficiently small magnitude, τ_A can be used to recover the unique (s, e) such that $As + e = r$ for all but a negligible fraction of possible r. Then one can use τ_A to efficiently recover all the preimages to any distribution Y in the image of the function η_κ (provided with a sample y from the distribution Y), except with negligible probability.*

Proof. Note, firstly, that κ_2 is either of the form $As + e$ for some (s, e) (with e of small enough magnitude), or it is not; there are no other alternatives. We do not know *a priori* which of these is the case, but the procedure that we perform in order to recover the preimage(s) to Y, given y, is the same in both cases:

1. Use the trapdoor τ_A to attempt to find (x_1, e_1) such that $Ax_1 + e_1 = y$. If such an (x_1, e_1) exists, record $0\|x_1$ as the first preimage.
2. Use the trapdoor τ_A to attempt to find (x_2, e_2) such that $Ax_2 + e_2 = y - \kappa_2$. If such an (x_2, e_2) exists, record $1\|x_2$ as the second preimage.

If $\kappa_2 = As + e$ for some s and e, then this procedure will (except with negligible probability) return two preimages. In step 1, it will recover x such that $y = Ax + e_0$ for some small e_0, because (by linearity) y is always of the form $Ax + e_0$. In step 2, it will recover $x' = x - s$, because $x' = x - s$ will satisfy the equation $y - (As + e) = Ax' + e'$ for $e' = e_0 - e$. We know that η_κ has two preimages when $\kappa_2 = As + e$, so this is all of the preimages to Y under η_κ and no others.

It can be seen by similar reasoning that, when $\kappa_2 = u$ for $u \neq As + e$, this procedure will return (except with negligible probability) exactly one preimage, which is what we expect when $\kappa_2 = u$. □

In the context of Protocol 3.3, the honest prover knows that η_{κ_i} has been evaluated correctly for all i, because the prover evaluated these functions for itself. Therefore, given Claim A.2, if our goal is to show that the honest prover can efficiently determine whether or not a purported trapdoor τ'_{A_i} can be used to recover all the preimages to y_i under η_{κ_i}, with $\kappa_i = (A_i, \kappa_{2,i})$, it is sufficient to show that a procedure exists to efficiently determine whether or not τ'_{A_i} truly 'inverts A_i', i.e. recovers (s, e) correctly from all but a negligible fraction of possible $r = A_i s + e$. Given the instantiation of τ_A presented in [MP11], this is simple to achieve: for example (using notation from Algorithm 1 of the same paper), the prover can request that the verifier sends it $\bar{\mathbf{A}}, \mathbf{H}$ and $\mathbf{R}$—$\mathbf{G}$ is public—and check whether $A = [\bar{\mathbf{A}} \mid \mathbf{HG} - \bar{\mathbf{A}}\mathbf{R}]$.

B Completeness and soundness of Protocol 2.4

For notational convenience, define $\alpha = \frac{a}{\sum_s 2|d_s|}$ and $\beta = \frac{b}{\sum_s 2|d_s|}$.

The Chernoff bound for Bernoulli variables gives us

$$Pr\Big[|\textsc{Count} - \mu| \geq \delta\mu\Big] \leq 2e^{-\frac{\mu\delta^2}{3}}.$$

We would like to calculate the probability that the deviation $|\textsc{Count} - \mu|$ is greater than $\frac{1}{2}m(\beta - \alpha)$, with $\mu = m(\frac{1}{2} - \alpha)$, because this is an upper bound for the probability that we will make the wrong decision when we are given the genuine witness $\rho = \sigma^{\otimes m}$. We proceed to calculate the appropriate δ.

$$\begin{aligned}
\delta\mu &= \frac{1}{2}m(\beta-\alpha)\\
\delta &= \frac{1}{2}(\beta-\alpha)\cdot\frac{m}{\mu}\\
\implies \frac{\mu\delta^2}{3} &= \frac{1}{12}(\beta-\alpha)^2\cdot\frac{m^2}{\mu}\\
&= \frac{1}{12}(\beta-\alpha)^2\cdot\frac{m}{\frac{1}{2}-\alpha}.
\end{aligned}$$

Since $\beta-\alpha$ is inverse polynomial, by [MNS16], and α is bounded away from $\frac{1}{2}$, the probability $2e^{-\frac{\mu\delta^2}{3}}$ can be made exponentially small by choosing m to be a sufficiently large constant times $|x|(\beta-\alpha)^2(\frac{1}{2}-\alpha)^{-1}$. The completeness of Protocol 2.4 follows.

To prove soundness, we simply replace μ with $m(\frac{1}{2}-\beta)$ in the calculation above, and once again arrive at the conclusion that the probability we make the wrong decision can be made exponentially small by choosing m as above, with $(\frac{1}{2}-\alpha)^{-1}$ replaced by $(\frac{1}{2}-\beta)^{-1}$.

C Commitment scheme

We provide an informal description of a generic form for a particular (and commonly seen) kind of commitment scheme. The protocol for making a commitment under this scheme requires three messages in total between the party making the commitment, whom we refer to as the *committer*, and the party receiving the commitment, whom we call the *recipient*. The first message is an initial message i from the recipient to the committer; the second is the commitment which the committer sends to the recipient; and the third message is a reveal message from the committer to the recipient. The scheme consists of a tuple of algorithms $(\mathsf{gen}, \mathsf{initiate}, \mathsf{commit}, \mathsf{reveal}, \mathsf{verify})$ defined as follows:

- $\mathsf{gen}(1^\ell)$ takes as input a security parameter, and generates a public key pk.
- $\mathsf{initiate}(pk)$ takes as input a public key and generates an initial message i (which the recipient should send to the committer).
- $\mathsf{commit}(pk, i, m, s)$ takes as input a public key pk, an initial message i, a message m to which to commit, and a random string s, and produces a commitment string z.
- $\mathsf{reveal}(pk, i, z, m, s)$ outputs the inputs it is given.
- $\mathsf{verify}(pk, i, z, m, s)$ takes as argument an initial message i, along with a purported public key, commitment string, committed message and random string, evaluates $\mathsf{commit}(pk, i, m, s)$, and outputs 1 if and only if $z = \mathsf{commit}(pk, i, m, s)$.

For brevity, we sometimes omit the public key pk and the initial message i as arguments in the body of the paper. The commitment schemes which we assume to exist in the paper have the following security properties:

- *Perfectly binding*: If $\mathsf{commit}(pk, i, m, s) = \mathsf{commit}(pk, i, m', s')$, then $(m, s) = (m', s')$.
- *(Quantum) computationally concealing*: For any public key $pk \leftarrow \mathsf{gen}(1^\ell)$, fixed initial message i, and any two messages m, m', the distributions over s of $\mathsf{commit}(pk, i, m, s)$ and $\mathsf{commit}(pk, i, m', s)$ are quantum computationally indistinguishable.

It has been stated informally ([?], [?]) that a commitment scheme with the above form and security properties can be obtained from a quantum-secure pseudorandom generator (the latter of which exists assuming the quantum hardness of LWE).

References

[ABE10] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations. In Andrew Chi-Chih Yao, editor, *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 453–469. Tsinghua University Press, 2010.

[ABOEM17] Dorit Aharonov, Michael Ben-Or, Elad Eban, and Urmila Mahadev. Interactive proofs for quantum computations. *arXiv preprint arXiv:1704.04487*, 2017.

[BFK09] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 517–526. IEEE, 2009.

[BJSW16] Anne Broadbent, Zhengfeng Ji, Fang Song, and John Watrous. Zero-knowledge proof systems for qma. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 31–40. IEEE, 2016.

[BL08] Jacob D. Biamonte and Peter J. Love. Realizable Hamiltonians for universal adiabatic quantum computers. *Physical Review A*, 78:012352, July 2008.

[FK17] Joseph F Fitzsimons and Elham Kashefi. Unconditionally verifiable blind quantum computation. *Physical Review A*, 96(1):012303, 2017.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, July 1991.

[Mah18] Urmila Mahadev. Classical verification of quantum computations. In *Foundations of Computer Science (FOCS), 2018 IEEE 59th Annual Symposium on*, pages 259–267, Oct 2018.

[MF16] Tomoyuki Morimae and Joseph F. Fitzsimons. Post hoc verification with a single prover. *arXiv e-prints*, March 2016. https://arxiv.org/pdf/1603.06046.pdf.

[MNS16] Tomoyuki Morimae, Daniel Nagaj, and Norbert Schuch. Quantum proofs can be verified using only single-qubit measurements. *Physical Review A*, 93:022326, February 2016.

[MP11] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. Cryptology ePrint Archive, Report 2011/501, 2011. https://eprint.iacr.org/2011/501.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

[RUV13] Ben W Reichardt, Falk Unger, and Umesh Vazirani. Classical command of quantum systems. *Nature*, 496(7446):456, 2013.

[Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM Journal on Computing*, 39(1):25–58, 2009.