Quantum Verification

Danya Lette, Jonah Macan, Armanpreet Pannu

December 2019

Contents

1 Introduction

One of the driving forces behind the field of quantum computing is the prospect of speedups: some problems that are intractable on classical computers might be solved efficiently on quantum computers. A significant body of research is devoted to identifying such problems and designing quantum algorithms for solving them. However, any progress we make towards this goal is of limited use until we can answer this question: How do we know when we can trust a quantum computer? In particular, if we are using quantum computers to solve problems that are hard for classical computers, then how can we determine classically that the quantum computers are giving us the correct result?

In this paper, we start by explaining some of the concepts that are required in order to formalize this question. Then, we give an overview of Mahadev's 2018 positive result. Finally, we describe two cutting-edge applications of quantum verification.

2 Background

2.1 Verification

Verification consists of proving that a program or algorithm is correct, i.e. that the output of the program matches a specification. Classically, a number of tools and techniques have been developed to aid in verification. Sometimes, verification simply consists of a pen-and-paper proof that an algorithm will, in theory, produce the correct result. Verification may also check that the actual result of a computation has a desired property; this form of verification is called *post-hoc*.

Many classical techniques are of limited use in the realm of quantum computing, due to a number of factors, including: the computational complexity of the problems that are of interest in quantum computing, the probabilistic nature of quantum, the difficulties posed by quantum measurement, and the unreliability of the existing quantum hardware. Proving that an algorithm is correct in theory is not an adequate form of verification, given that the algorithm will likely be executed on extremely noisy hardware. On the other hand, a post-hoc verification procedure has to contend with the fact that the result of the computation is a quantum state and cannot easily become known to a classical verifier, let alone verified.

The central question in quantum verification is whether classically intractable problems must have classically intractable verification procedures. This question is made more precise in the following sections by the introduction of the concept of interactive proof systems, and the definition of several complexity classes.

2.2 Self-test

Quantum verification should be distinguished from device-independent quantum self-test, also known as state tomography. Self-test is concerned with ascertaining which state a device is outputting, without having knowledge of – or access to – the physical properties of the quantum device [1]. One application of self-test is determining whether the output state of a quantum device is entangled. In the device-independent scenario, a quantum device is treated as black box rather than as a specific laboratory setup in which we might be able to determine or manipulate the device directly using specific hardware (such as by applying optical or microwave pulses). Nonetheless, self-test does play an important role in some quantum verification protocols, such as [2].

2.3 Soundness and Completeness

The usefulness and trustworthiness of a proof system may be evaluated in a variety of ways. Ideally, a proof system is both sound and complete. We say that a proof system PS is sound if, whenever we can construct a proof of a claim C, then C is valid. In other words, if C isn't valid, then you can't use PS to prove C. A proof system PS is complete if every valid claim has a PS-proof. The evaluation of a proof system PS can be more nuanced. For example, we might be interested in the probability of a claim's validity if that claim has a PS proof, the PS-provability of claims that have certain properties, etc.

2.4 Interactive Proof System

An Interactive Proof System (IPS) models a computation as an interactive protocol in which there are (usually) two parties – a prover and verifier. IPS can be thought of as a formalization of the notion that a proof consists of one party convincing another party that they are correct. When modelling a simple decision problem using an IPS, we can think of the prover as having the job of deciding YES or NO and of sending the verifier some carefully chosen information called a *witness* which constitutes proof of its decision. The verifier then uses the witness to ascertain if the prover's decision is correct. An IPS may also involve several rounds of communication between the verifier and the prover.

Modelling a computation as an IPS is useful because it separates a computation into discrete parts and thus gives us a formal way of discussing properties of its parts. For instance, the computational complexity of the prover may differ from that of the verifier. The number of messages that the prover and verifier use to communicate may also be categorized in terms of computational complexity. Importantly, we may choose to implement the prover on a quantum computer, and the verifier on a classical computer. This makes IPS a natural choice of proof system in quantum computing.

However, it's not obvious whether every computational problem admits an IPS verification protocol. Also, remark that any proof system will only be useful to us if it can be implemented efficiently for the problems that are of interest in quantum computing. Given these restrictions, it is not obvious *prima facie* that IPS can serve as general-purpose verification systems in QC. These concerns are formalized as questions about the relationship between a set of complexity classes.

2.5 Complexity Classes

2.5.1 BPP

BPP is the class of bounded-error probabilistic polynomial time decision problems that can be solved in polynomial time by a probabilistic Turing Machine with an error bounded by $\frac{1}{3}$. Note that $P \subseteq BPP$.

2.5.2 IP

Interaction Polynomial is the class of problems solvable by an interactive proof system with a BPP verifier (i.e. a polynomial verifier that has access to a random number generator and whose error is bounded by $\frac{1}{3}$) and in which the prover and verifier share a polynomial number of messages.

2.5.3 MA

Merlin-Arthur is the same as IP except that the prover and verifier exchange only one message. In addition, the verifier must accept with probability $\geq \frac{2}{3}$ on a YES instance, and must accept with probability $\leq \frac{1}{3}$ on a NO instance.

2.5.4 Quantum Complexity

The quantum analogues of the above classes are BQP, QIP, and QMA. QMA is interesting because $P, BQP, NP \subseteq QMA$. In addition, the local Hamiltonian problem is QMA-complete [3], so we may use reductions to the local Hamiltonian problem to solve many other interesting problems.

2.6 The Question

Using the terms we have just defined, we are now in a position to formalize the question "How do we know when we can trust a quantum computer?" There are many different approaches but the formulation that has gained the most traction is this: Does every language in the class BQP admit an interactive protocol where the prover is in BQP and the verifier is in BPP? [1]

3 The Mahadev Paper

In 2018, a UC Berkley PhD student named Urmila Mahadev proposed a mechanism for doing efficient classical verification of an efficient quantum computation [5]. More precisely, she proposed a protocol for a sound and complete interactive proof system in which the prover is BQP and the verifier is BPP. It was previously shown [6] that a BPP verifier can verify a BQP prover if the verifier is given access to a trusted (quantum) measurement device. This is not the ideal scenario insofar as this requires that the verifier is either not classical, or is classical but has access to two quantum devices – the prover as well as a measurement device. The central innovation of Mahadev's approach is that her prover can serve as its own trusted measurement device.

Recall that BQP is concerned with decision problems; some inputs to a given decision problem are YES instances while others are NO instances. We can define a set L such that $x \in L$ whenever x is a YES instance. For example, $(x_1 \vee x_2) \in SAT$, the language of satisfiability, since the formula $(x_1 \vee x_2)$ is satisfiable and thus is a YES instance of SAT. For a language $L \in BQP$, and instance x, we want an algorithm which determines if $x \in L$. Mahadev's verifier performs the following steps:

Verifier Protocol

VP1. Reduce x to a local Hamiltonian H_x

VP2. Request that the prover generate an n-qubit quantum state – this state is akin to a "witness" and should correspond to the ground state of H_x

VP3. Perform a measurement of the state.

VP4. Using the results of the measurement, check if the state has low energy. If so, then $x \in L$. If not, then $x \notin L$

[1]This question was posed in a blog post by Scott Aaronson in 2006 [4], in which he offered a substantial $25.00 prize to whoever could come up with an answer.

This is a fairly straightforward protocol for an interactive proof system. However, VP3 is actually quite complicated to perform, since the verifier is classical and thus cannot directly perform quantum measurements. This step is what Mahadev calls the *measurement protocol*, and is the central focus of the paper. I will briefly explain each of these steps in the following section.

3.1 The Protocol

3.1.1 VP1

VP1. Reduce x to a local Hamiltonian H_x

In this step, the verifier reduces the instance x to a local Hamiltonian. We are assured that this reduction is possible in general because $BQP \subseteq QMA$, and the local Hamiltonian problem is QMA-complete [3]. Furthermore, the reduction procedure can be done by a BPP machine.

This reduction is desirable because any problem in BQP can be reduced to a local Hamiltonian problem. This allows us to construct a general verification procedure (i.e. one that applies to all problems in BQP) by analysing a specific problem.

3.1.2 VP2

VP2. Request that the prover generate an n-qubit quantum state – this state is akin to "witness" and should correspond to the ground state of H_x

Since the prover is more or less a black box, we are not concerned with how the prover finds this state – it is enough for us to know that it can find it, since it is a BQP machine, and local Hamiltonian is in QMA.

3.1.3 VP3

VP3. Perform a measurement of the state.

In the previous step, the verifier asks the prover for a quantum state. This state does not need to be communicated to the verifier.[2] Rather, we ask the prover to measure its own state. By augmenting this step with a cryptographic protocol called a commitment scheme, the prover is "committed" to measuring the state that it generated in the previous step. This allows the verifier to be assured that prover does not modify this choice at a later stage.

After making its choice, the prover sends the verifier some information that we call a commitment. The commitment is computed by the prover by applying a procedure to the chosen state. The procedure that is used to compute the commitment must have several interesting properties in order to satisfy our ultimate goal, which is to force the prover to make trustworthy measurements of its own states. This requirement motivated Mahadev to define two families of functions, the Extended Trapdoor Claw-Free family and the Trapdoor Injective family. Let $\mathcal{F}$ be such a family, where $\mathcal{F} : \{f_{k,0} : \mathcal{X} \to \mathcal{Y}, f_{k,1} : \mathcal{X} \to \mathcal{Y}\}$ and $\mathcal{X} = \{0,1\}^w$.

[2] In fact, since the verifier is classical, it is not even possible for it to receive this state. This detail sets Mahadev's IPS protocol apart from a conventional IPS, since typically the witness is simply sent to the verifier.

Verifier Measurement Protocol

VM1. Choose an n-qubit string $h = (h_1, ..., h_n) \in \{0,1\}^n$ (where the value of the ith coordinate determines whether to make a Hadamard or standard basis measurement on the ith coordinate of the witness)

VM2. Choose key bits $k_1, ..., k_n$ and trapdoor bits $t_1, ..., t_n$ corresponding to the family of trapdoor functions, and send the keys to the prover. (The ith function should be a Trapdoor Claw-Free function if $h_i = 0$, and Trapdoor Injective otherwise.)

VM3. Ask prover for a classical commitment $y_1, ..., y_n$ to the witness

VM4. Choose at random whether to run a test round or a Hadamard round

VM5. Depending on that choice,

(a) If test round was selected, the verifier requests standard basis measurements of the committed qubit and preimage register for all n qubits.

(b) If Hadamard round was selected, the verifier requests Hadamard basis measurements of the committed qubit and preimage register for all n qubits.

VM6. Decode the measurement result using the trapdoor and check the result against the commitment y_i. If this check fails for any bit, reject. Otherwise, accept.

Prover Measurement Protocol (Claw-Free Case)

PM1. Receive the keys $k = \{k_1, ..., k_n\}$ from the verifier.

PM2. For each single qubit state $|\psi_i\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ and corresponding key bit k_i, let $f_{k_i,0}, f_{k_i,1}$ from the requisite function family. (If $h_1 = 0$, these functions should come from the claw-free family. Otherwise, they should come from the injective family. For simplicity, I assume all are the former.)

PM3. Apply $f_{k_i,0}/f_{k_i,1}$ to a uniform superposition of $x \in \mathcal{X}$ over the input to form the state $\frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \alpha_0 |0\rangle |x\rangle |f_{k_i,0}(x)\rangle + \alpha_1 |1\rangle |x\rangle |f_{k_i,1}(x)\rangle$

PM4. Measure the final register to obtain $\alpha_0 |0\rangle |x_{0,y_i}\rangle |y_i\rangle + \alpha_0 |0\rangle |x_{0,y_i}\rangle |y_i\rangle$

PM5. Send the y_is to the verifier. This $y = \{y_1, ..., y_n\}$ is the commitment string. Remark that these are classical bits, since they are the result of making a measurement on that register.

PM6. If the verifier requests a test round, measure and send the verifier the result. Otherwise, apply a Hadamard gate to both remaining registers and send the verifier the result.

I have included the measurement protocol out of interest; However, the details of this scheme are beyond the scope of this paper. The main gist of the protocol is that the verifier gains leverage over the prover by utilizing a function that only she can efficiently invert.

3.1.4 VP4

VP4. Using the results of the measurement, check if the state has low energy. If so, then $x \in L$. If not, then $x \notin L$

Recall that we began the protocol by constructing H, and that the committed state ρ is intended to be a ground state of H. We would like to know its energy and thus to perform $H\rho$.

Mahadev's analysis of how a classical verifier can accomplish this feat is taken largely from [6]. To start, the verifier randomly chooses H_i, one of the local Hamiltonians of H. From [7], we can assume that H is 2-local and only has X and Z operators, since this model of Hamiltonians is QMA-complete. Furthermore, applying X and Z operators is equivalent to performing measurement in a chosen basis. From these facts, and since H_i only acts non-trivially on 2 qubits, the verifier may, in theory, approximate the energy of ρ simply by measuring a couple of its bits. However, the verifier does not perform these measurements. In fact, these steps were accomplished previously, during the measurement phase, when the verifier selected the measurement basis $h = (h_1, ..., h_n)$. The product of the resulting bits approximates the energy of $H\rho$. So, in this stage, the verifier is merely computing a product, and accepting x (i.e. concluding that $x \in L$) if the result is sufficiently small.

3.2 Soundness and Completeness

The protocol that Mahadev proposed may appear to be unnecessarily complicated. However, the details are necessary in order to ensure that it is sound and complete. As explained earlier, a general proof system is sound if it accepts *only* inputs that are valid. It is complete if it accepts *all* inputs that are valid. This definition fails to capture that probabilistic nature of quantum systems. We can account for that by rephrasing those criteria using the language of probability distributions.

Mahadev describes the soundness and completeness criteria as follows: For prover $\mathbb{P}$, basis choice $h = (h_1, ..., h_n)$, and an n qubit state ρ, let $D_{\mathbb{P},h}$ be the distribution of measurement results obtained by the verifier, and $D_{\rho,h}$ the distribution obtained by measuring ρ in basis h. We can think of $D_{\mathbb{P},h}$ as the output of an algorithm which computes $f(\rho)$ for some function f, and of $D_{\rho,h}$ as the true value of $f(\rho)$. The protocol is deemed complete if there exists a prover $\mathbb{P}$ such that $D_{\mathbb{P},h}$ is close enough to $D_{\rho,h}$ for all h, and $\mathbb{P}$ is accepted by the verifier with negligible error. The protocol is deemed sound if, whenever the verifier accepts $\mathbb{P}$, there exists a ρ such that $D_{\mathbb{P},h}$ is close enough to $D_{\rho,h}$.

The completeness proof of the protocol is quite simple - if the prover is honest, then the output of the algorithm is as desired. This follows directly from the construction of the protocol. The proof of soundness, however, takes up many pages of Mahadev's paper. These proofs demonstrate the correctness of usefulness of the protocol; without them, the protocol itself has little value.

4 Application: Computing the Order of a Group

4.1 Background

In this section, we present a concrete example of using an interactive proof system in which a verifier computes the value of a function f using the help of a more powerful and possibly unreliable prover. The verifier and prover are given an input x and, using the help of the prover, must output $f(x)$ or *null* with high probability if the verifier believes the prover is dishonest. We will first define the notion of computing f with an interactive prove system. This is similar to definitions presented above.

Definition 1. *Given a function $f : X \to \{0,1\}^*$, we say f has a interactive proof system if given* *verifier V and prover P who are both given an input $x \in X$ and are allowed to communicate via classical polynomial length messages, the verifier outputs $y \in \{0,1\}^*$ or null s.t:*

1. *There exists a prover P s.t. $y = f(x)$ with probability greater than $\frac{2}{3}$*
2. *Given any other prover P', $y = f(x)$ or null with probability greater than $\frac{2}{3}$*

Part 1 of this definition is the completeness condition and in this case, the prover is said to be an honest prover. Part 2 is the soundness condition and the prover is said to be dishonest. For this example we consider the problem of determining the order of solvable black-box group G using an interactive prove system created by Gall et al [8]. We assume the reader is familiar with basic notions in group theory and present some preliminary definitions relevant to the problem.

Definition 2. *A black-box representation for a group G is one in which the elements of G are represented by a n-bit string and whose group operations are performed efficiently by an oracle.*

In general we assume the number of bits n required to represent every element of G is $O(log(|G|)$. Given a black-box group representation of G, determining the order of G has been shown to be difficult in general on classical computers, even when the group is abelian. In a quantum system, a black box group has its elements represented by a n-qubit quantum state $|g\rangle$ and there exists oracles U and V s.t $U|g\rangle|h\rangle = |g\rangle|gh\rangle$ and $V|g\rangle = |g^{-1}\rangle$. Using quantum computers, the order of a quantum black-box group G can be determined efficiently for groups which are solvable.

Definition 3. *A group G is said to be solvable if there exist a generating sequence $\{h_1, ..h_n\} \subset G$ s.t. $G = \langle h_1, ..., h_n\rangle$ and $\langle h_1, ..., h_{i-1}\rangle \trianglelefteq \langle h_1, ..., h_i\rangle \forall i$. We refer to the sequence $\{h_1, ...h_n\}$ as a polycyclic generating sequence for G. For the remainder of this paper, we denote $H_i = \langle h_1, ..., h_i\rangle$*

Two relevant properties of solvable groups that we will use in this paper are:

1. If $h \in H_i$, then there exist a decomposition $h = h_1^{j_1}...h_i^{j_i}$ for some $j_1, ..., j_i \in \mathbb{Z}$
2. $|H_i| = |h_1|...|h_i| = |H_n/H_{n-1}|...|H_1/H_0|$

Theorem 1. *[9] Given a solvable black-box group G, there exists a quantum algorithm that runs in time $poly(log|G|)$ and outputs $|G|$ with probability $1 - \frac{1}{poly(|G|)}$*

We omit the outline of the algorithm and focus instead on a interactive proof system to compute $|G|$ in the case where we may have an unreliable quantum computer with a reliable classical verifier. To simplify the problem for the sake of example, we also assume that we are given the prime factors of $|G|$ as a set $S = p_1, ...p_l$. (Note this assumption may be omitted in exchange for a slightly more complicated IPS). We start by presenting algorithms that the honest quantum prover will need to perform.

Theorem 2. *[9] Let G be a black-box solvable group with $\{h_1, ..., h_t\}$ a polycyclic generating sequence of G. Then there exists two quantum algorithms A_1 and A_2 that run in time $poly(log|G|)$ s.t:*

1. *A_1 takes inputs integer $i < t$ and $h \in H_i$ and outputs the decomposition $h = h_1^{j_1}...h_i^{j_i}$ with probability $1 - \frac{1}{poly(|G|)}$*
2. *A_2 takes inputs integer $i < t$ and $h \in G$ and decides if $h \in H_i$ or not with probability $1 - \frac{1}{poly(|G|)}$*

In addition, there are a few actions that the classical verifier will need to perform:

Theorem 3. *[10] Let G be a black-box group and $\epsilon < 0$. There exists a classical algorithm running in time $log(|G|)$ and $log(\frac{1}{\epsilon})$ that outputs a random element $g \in G$ with probability p_g where $p_g \in (\frac{1}{|G|} - \epsilon, \frac{1}{|G|} + \epsilon)$*

Theorem 4. *[8] Let G be a black-box solvable group and $S = p_1, ...p_l$ be the set of prime factors of $|G|$. Then there exists a classical algorithm that runs in $poly(log|G|)$ and outputs elements $h_1, ..., h_t \in G$ and prime numbers $r_1, ...r_t \in S$ s.t. with probability $1 - \frac{1}{poly(|G|)}$ we have:*

1. *$\{h_1, ..., h_t\}$ is a polycyclic generating sequence of G*
2. *$|H_i/H_{i-1}| = 1$ or r_i*

4.2 The IPS protocol

Given the existence of these algorithms, an interactive proof system to compute $|G|$ is outlined as follows [8]:

1. The verifier uses theorem 4 to generate a polycyclic generating sequence $h_1, ..., h_t \in G$ and prime numbers $r_1, ...r_t \in S$.
2. For each i, the verifier randomly chooses $s_i \in \{0, 1\}$ and a random element $x_i \in H_{i-1}$ using theorem 3 by taking $\epsilon = \frac{1}{2^{2n}}$ where n is the number of qubits used to represent the elements of G
3. The verifier sends $h_1, ..., h_t$ and $h_i^{s_i} x_i$ for each i to the prover
4. The (honest) prover uses theorem 2 and uses algorithm A_1 to compute a_i where $a_i = 0$ if $h_i^{s_i} x_i \in H_{i-1}$ and $a_1 = 1$ else. If $h_i^{s_i} x_i \in H_{i-1}$, the prover uses A_2 to compute the decomposition of $h_i = h_1^{j_{i,1}} ... h_{i-1}^{j_{i,i-1}}$, else the prover randomly assigns values to $\{j_{i,k}\}$. The prover sends $a_1, ..., a_t$ and $\{j_{i,k}\}$ to the verifier.
5. For each i, the verifier sets l_i according to the following rules:
   (a) if $h_i = h_1^{j_{i,1}} ... h_{i-1}^{j_{i,i-1}}$ set $l_i = 1$
   (b) if $h_i \neq h_1^{j_{i,1}} ... h_{i-1}^{j_{i,i-1}}$ and $a_i = s_i$ set $l_i = r_i$
   (c) else verifier aborts the algorithm and outputs *null*
6. If the output is not *null* from step 5, the verifier outputs $|G| = \prod_i l_i$

We first show that this protocol outputs $|G|$ with high probability if the prover is honest in step 4 of the protocol. For each i, we know $|H_i/H_{i-1}| = 1$ or r_i. First suppose $|H_i/H_{i-1}| = 1$, then $h_i \in H_{i-1}$ and $H_i = H_{i-1}$ so regardless of the value of s_i, $h_i^{s_i} x_i \in H_{i-1}$. The honest prover, with high probability, will then return the correct decomposition $h_i = h_1^{j_{i,1}} ... h_{i-1}^{j_{i,i-1}}$ and the verifier will set $l_i = 1 = |H_i/H_{i-1}|$ as in rule $5(a)$.

On the other hand, if $|H_i/H_{i-1}| = r_i$, then $h_i \notin H_{i-1}$. In this case, if $s_i = 0$ (resp. $s_i = 1$) then $h_i^{s_i} x_i \in H_{i-1}$ (resp. $h_i^{s_i} x_i \notin H_{i-1}$) and the honest prover, with high probability, will output $a_i = s_i$ and a random incorrect decomposition since no correct decomposition exists. According to step $5(b)$, the verifier will set $l_i = r_i = |H_i/H_{i-1}|$. In the last step, the verifier will output $\prod_i l_i = \prod_i |H_i/H_{i-1}| = |G|$ with a high probability as desired.

Now we analyze the case where we don't have an honest prover in step 4. The protocol will output a value different from $|G|$ or *null* if and only if the prover convinces the verifier to set $l_i \neq |H_i/H_{i-1}|$ from some i in step 5. If $|H_i/H_{i-1}| = r_i$, the dishonest prover can not provide a correct decomposition $h_i = h_1^{j_{i,1}}...h_{i-1}^{j_{i,i-1}}$ so the verifier will never incorrectly set $l_i = 1$ according to $5(a)$. In the case where $|H_i/H_{i-1}| = 1$, the verifier will incorrectly $l_i = r_i$ only if the prover outputs $a_i = s_i$. This means the dishonest prover must guess correctly if $h_i^{s_i}x_i \in H_{i-1}$. The probability of guessing correctly is:

$$
\begin{aligned}
p_i &= \frac{1}{2} + \frac{1}{2}\sum_{h\in H_{i-1}} \left|\frac{1}{2}(Pr[x_i = h] - Pr[h_i x_i = h])\right| \\
&= \frac{1}{2} + \frac{1}{4}\sum_{h\in H_{i-1}} \left|Pr[x_i = h] - \frac{1}{|H_{i-1}|} - Pr[h_i x_i = h] + \frac{1}{|H_{i-1}|}\right| \\
&\leq \frac{1}{2} + \frac{1}{4}\sum_{h\in H_{i-1}} \left|Pr[x_i = h] - \frac{1}{|H_{i-1}|}\right| - \frac{1}{4}\sum_{h\in H_{i-1}} \left|Pr[h_i x_i = h] - \frac{1}{|H_{i-1}|}\right| \\
&= \frac{1}{2} + \frac{1}{2}\sum_{h\in H_{i-1}} \left|Pr[x_i = h] - \frac{1}{|H_{i-1}|}\right| \\
&\leq \frac{1}{2} + \frac{1}{2}\sum_{h\in H_{i-1}} \epsilon \\
&= \frac{1}{2} + \frac{1}{2}|H_{i-1}|\epsilon \\
&\leq \frac{1}{2} + \frac{1}{2}2^n\frac{1}{2^{2n}} \\
&= \frac{1}{2} + \frac{1}{2^{n+1}}
\end{aligned}
$$

Above the third line used the triangle inequality, the fourth line uses that since $h_i \in H_{i-1}$, $h_i^{-1}h \in H_{i-1}$ and $Pr[h_i x_i = h] = Pr[h_i = h_i^{-1}h]$, the two sums are the same. The fifth line uses theorem 3 and the sixth line uses that n is the encoding length of G.

Thus with probability $\frac{1}{2} + \frac{1}{2^{n+1}}$, the prover will guess correctly if $a_i = s_i$. Recall the verifier will output a value different from $|G|$ or *null* only if verifier incorrectly sets $l_i = r_i$ for some i. The above analysis shows that this has a probability less than $\frac{1}{2} + \frac{1}{2^{n+1}}$. The probability that the verifier will output $|G|$ or *null* is thus $\frac{1}{2} - \frac{1}{2^{n+1}}$. By running this algorithm in parallel, m times reduces this probability to $1-(\frac{1}{2}+\frac{1}{2^{n+1}})^m$ which allows us to reach the threshold of $\frac{2}{3}$ in definition 1.

In conclusion, this algorithm provides a interactive proof system in which a reliable classical verifier may use a possibly unreliable quantum prover to compute the order of a group or detect a unreliable prover and output *null* with a high probability.

5 Application: Blind Quantum Computing

We now turn our attention to a more practical application of quantum verification, namely Blind Quantum Computing. Blind computing is used for doing computation on encrypted data which could contain sensitive information, such as financial or health records, while preserving personal privacy and information security. As in the verification procedures described in the previous sections, this procedure involves two parties. We call the verifier Alice or the user, and the prover Bob or the server. Alice asks Bob to execute a quantum program and report the measurements back to Alice in such a way that Alice can check the validity or 'honesty' of the quantum computations. In its most basic form, the Alice applies an encryption procedure E to input x. Bob computes output z which is sent to Alice, who then applies a decryption procedure $D(z)$ to obtain the desired output [11], [12].

In the quantum domain, we assume that Alice can implement BPP computations but requires Bob to carry out BQP computations while keeping the input, the output, and the function of interest (the one acting on the input) hidden from the Bob. The first solution to this was given by Childs' Protocol, developed by Andrew Childs, computer scientist and physicist at the University of Maryland [13].

5.1 Setup

In this protocol, Alice has the ability to perform basic quantum gates, but is unable to implement arbitrary quantum circuits. This differs from the protocols described in previous sections, in which the verifiers were purely classical. We assume Alice can generate random classical bits, store quantum states, prepare only the $|0\rangle$ state, perform the SWAP gate (which sends $|01\rangle$ to $|10\rangle$ and vice-versa), and Pauli Gates (namely X, Z, and their product XZ). However, she may not perform measurements. Bob, on the other hand, can perform arbitrary quantum measurements and can do universal quantum computation. Lastly, this protocol allows for bidirectional quantum communication between Alice and Bob.

5.2 Idea

The protocol uses a quantum version of the classical Vernam cipher, also known as a one-time pad. In the classical setting, Alice wishes to send a bit b to Charlie and both share a bit k, known as the private key. Alice sends the message $m = b \bigoplus k$ (where $\bigoplus$ represents addition mod 2). Then Charlie uses k to decrypt m and recover b. In a private quantum channel on the other hand, we still use a classical key k, but the bit b we wish to send now becomes a quantum state $|\Psi\rangle$. Since each qubit is characterized by two numbers, the key is now comprised of two classical bits, k and j. In this scenario, Alice applies $Z^k X^j$ to $|\Psi\rangle$. If a third-party intercepts this, they won't be able to decipher the original qubit without knowledge of j and k. Presuming they chose to leave the qubit alone, Charlie can then decode the qubit by applying $X^j Z^k$ to reverse the operations and recover $|\Psi\rangle$.

5.3 Implementation

The implementation of Childs' protocol requires two main steps:

Problem 1: How can Bob help Alice measure states in the computational basis?

Alice chooses random bits j and k and applies the unitary $Z^k X^j$ to her state. Once Bob receives the qubit, measures it, and reports the result back to Alice, she can recover the original state by simply flipping the qubit if $j = 1$ and do nothing otherwise (since applying Z will not change the measurement outcomes). This process is depicted in the circuit below where the double lines represent the shared, random classical bits and a dashed box around a gate represents the result of an 'honest' computation, or measurement by Bob.

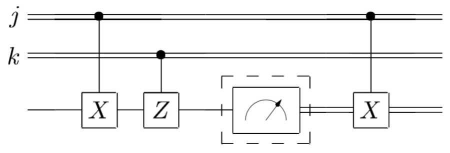

Figure 1: Secure-assisted computational basis measurement

Problem 2: How can Bob help her extend her initial gate set to perform universal quantum computations?

The protocol provides circuits for implementing the Hadamard, CNOT, and $\frac{\pi}{8}$ (or T) gates, which constitute a universal gate set.

(I) For the Hadamard gate depicted below, Alice applies $Z^k X^j$ to her qubit. Provided Bob is honest and performs the Hadamard gate, Alice can recover the original qubit with just the H gate applied to it by reversing X by Z and Z by X. This follows from the fact that $XHZ = ZHX = H$.

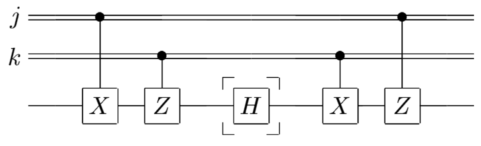

Figure 2: Implementation of the Hadamard gate

(II) For the CNOT gate shown below, Alice requires four random (classical) bits j, k, l, and m since CNOT requires two qubits to function. She applies $Z^k X^j$ to the first qubit and $Z^m X^l$ to the second qubit. Provided Bob is honest and performs the CNOT gate faithfully, Alice can decode the returned qubits based on what here initial classical bits were.

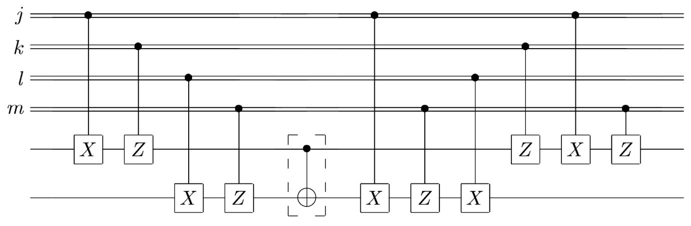

Figure 3: Implementation of the CNOT Gate

(III) For the $\frac{\pi}{8}$ gate reproduced below, Alice needs a two-round protocol, unlike the above two (this arises from the fact that X does not commute with T). Starting again with four random (classical bits) j, k, l, and m, Alice randomizes her state by applying $Z^k X^j$. If Bob performs the T gate honestly and returns the result to Alice, she can apply $X^j Z^k$ to the output. Z commutes with T ($ZT = TZ$) but for X we have that $XTX = T^*$ which differs from T by T^2 (since T is unitary $T^*TT = IT = T$). Now Alice will send this state back to Bob for calculation and this time, since $T^2 = Z$, Alice can reverse the randomization without any issue to recover the T gate calculation.

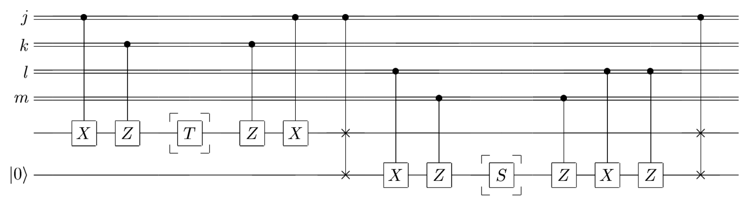

Figure 4: Implementation of the T Gate

5.4 Challenges

Although the protocol is effective at preventing Bob from obtaining information from Alice, it has a few weak spots that merit further discussion. For one thing, it does not prevent Bob from refusing to carry out Alice's computation or from Bob returning an erroneous calculation. These weaknesses are innate to the private quantum channel, but there are some partial solutions. In the case of NP problems, Alice has a way of keeping Bob honest by simply checking if the given output satisfies the initial problem (i.e., testing the output of two integers using Shor's Algorithm to see if they are indeed factors of the input number). However, if the problem is not in the NP complexity class, it may be difficult or impossible for Alice to verify Bob's output. In this case, assuming Bob is a memoryless black box, Alice could bound the probability Bob is cheating by feeding randomized subsets of the input and performing tests on the outputs. A final and more practical problem inherent in Childs' protocol is the use of a quantum channel to send an unrestricted number of quantum states back and forth at will. In a practical setting, such a channel may be too slow or expensive and thus we would require a more limited transfer of quantum states.

Overall, Childs' protocol provides an interesting and practical application of quantum verification for guaranteeing both the validity of quantum computations, while preserving the privacy and personal information of the user Alice.

6 Conclusion

In conclusion, we have explored the problem of quantum verification, culminating in an analysis of the Mahadev's 2018 paper, which proposes a sound and complete interactive proof system for doing efficient classical verification of an efficient quantum computation. The ability to tell whether a quantum computer is 'honest' has many important theoretical and practical applications, as shown in both Childs' protocol for blind quantum computation and in computing the order of a group. Quantum verification will no doubt play an important role in the future of computing.

References

[1] I. Supic, J. Bowles, Self-testing of quantum systems: a review, arXiv e-prints (2019) arXiv:1904.10042`arXiv:1904.10042`.

[2] A. Gheorghiu, E. Kashefi, P. Wallden, Robustness and device independence of verifiable blind quantum computing, New Journal of Physics 17 (8) (2015) 083040. arXiv:1502.02571, doi:10.1088/1367-2630/17/8/083040.

[3] J. Kempe, A. Kitaev, O. Regev, The Complexity of the Local Hamiltonian Problem, arXiv e-prints (2004) quant–ph/0406180arXiv:quant-ph/0406180.

[4] S. Aaronson, The Aaronson $25.00 Prize, Shtetl-Optimized: The Blog of Scott Aaronson.
URL https://www.scottaaronson.com/blog/?p=284

[5] U. Mahadev, Classical Verification of Quantum Computations, arXiv e-prints (2018) arXiv:1804.01082arXiv:1804.01082.

[6] T. Morimae, J. F. Fitzsimons, Post hoc verification with a single prover, arXiv preprint arXiv:1603.06046.
URL https://arxiv.org/pdf/1603.06046v1.pdf

[7] J. D. Biamonte, P. J. Love, Realizable hamiltonians for universal adiabatic quantum computers, Physical Review A 78 (1). doi:10.1103/physreva.78.012352.
URL http://dx.doi.org/10.1103/PhysRevA.78.012352

[8] F. L. Gall, T. Morimae, H. Nishimura, Y. Takeuchi, Interactive proofs with polynomial-time quantum prover for computing the order of solvable groups (2018). arXiv:1805.03385.

[9] J. Watrous, Quantum algorithms for solvable groups, in: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing, STOC '01, ACM, New York, NY, USA, 2001, pp. 60–67. doi:10.1145/380752.380759.
URL http://doi.acm.org/10.1145/380752.380759

[10] L. Babai, Local expansion of vertex-transitive graphs and random generation in finite groups, in: Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing, STOC '91, ACM, New York, NY, USA, 1991, pp. 164–174. doi:10.1145/103418.103440.
URL http://doi.acm.org/10.1145/103418.103440

[11] J. F. Fitzsimons, Private quantum computation: an introduction to blind quantum computing and related protocols, npj Quantum Information 3 (2017) 23. arXiv:1611.10107, doi:10.1038/s41534-017-0025-3.

[12] A. Gheorghiu, T. Kapourniotis, E. Kashefi, Verification of quantum computation: An overview of existing approaches, Theory of Computing Systems 63 (4) (2019) 715–808. doi:10.1007/s00224-018-9872-3.
URL https://doi.org/10.1007/s00224-018-9872-3

[13] A. M. Childs, Secure assisted quantum computation, arXiv e-prints (2001) quant–ph/0111046arXiv:quant-ph/0111046.