Lecture 4

Testing a qubit under computational assumptions

As we discussed in the first lecture, a simple interactive "test of quantumness" under computational assumptions consists in asking the device to factor a large integer n; under the assumption that factoring is hard for classical computers this test adequately distinguishes classical from quantum devices.

The main limitation of this test that is generally pointed out is that in order for a device to successfully demonstrate its "quantumness" it needs to have the capability to implement a large, fault-tolerant quantum computation. In contrast, the test based on spatial isolation that we saw in the previous lecture can be executed with a pair of two-qubit devices.[1]

A second limitation that is relevant for us is that the test does not seem to provide a means to certify a qubit. Modeling the prover in a "factoring test" would give us a family of POVM $\{\Pi^n_{\{p_i\}}\}$, indexed by integers n and whose outcomes are lists of primes $\{p_i\}$. In fact, for each n there should be a single POVM element —the one with the correct outcome— such that $\Pi^n_{\{p_i\}}|\psi\rangle = |\psi\rangle$ with $|\psi\rangle$ the initial state of the device; in somewhat informal notation that POVM element is supposed to be obtained as $U^\dagger|\{p_i\}\rangle\langle\{p_i\}|U$ with U a circuit implementation of Shor's algorithm. But to identify a qubit we know that we need *two* observables acting on the same space as well as some indication that these observables ought to be "incompatible" (anti-commutation). It is not at all clear how to identify such observables here.

A key insight from this lecture is that in order to obtain a computational test for a qubit we will need to assume that a certain problem is hard not for classical computers but also for quantum computers. This is because our model requires us to identify two observables X and Z in the device such that the device has the ability to perform either X and Z but not both simultaneously. In Lecture 2 this incompatibility arose from the necessity for X and Z to yield predictions that matched those of σ_X and σ_Z. In the second lecture, it arose from some form of information-theoretic impossibility—in a loose sense, had X and Z (recall that in the notation from the previous lecture these were identified as B_4 and B_2) been compatible, then the Magic Square would have had a classical solution—which it doesn't.

In this lecture the impossibility of measuring X and Z will be based on considerations of computational difficulty. We will show that, if the quantum device was able to measure X and Z jointly then it would break a computational problem that is assumed to be hard *even for quantum computers* — we will formulate this later as a form of "computational uncertainty principle". Since by definition the device can measure X

[1] The test does require the ability to distribute entanglement across a large distance—if one uses relativity to certify the assumption of spatial isolation—and to perform fairly high-fidelity measurements on it. This is not at all easy, but it can be done today; implementations of Shor's algorithm that outperform the best classical factoring algorithms are not expected within the next decade.

and Z separately, if they commuted then it could also measure them jointly. Therefore, the computational assumption gives rise to an *information-theoretic* consequence on the observables X and Z: they must form a qubit.

4.1 Simon's algorithm

As a warm-up we start by reminding ourselves of a typical example of a kind of task for which the manipulation of quantum information provides a computational advantage.

4.1.1 The algorithm

The input to an instance of Simon's problem is a function $f : \{0,1\}^n \to \{0,1\}^n$ that has the property that f is 2-to-1 (every value in the range has exactly two preimages) and moreover there is a string $s \in \{0,1\}^n$ such that for every $x, y \in \{0,1\}^n$, $f(x) = f(y)$ if and only if $y = x$ or $y = x + s$, where addition is performed coordinate-wise and modulo 2. The goal is to recover the string s. It is not hard to see that in the worst case any classical algorithm requires at least $\Omega(2^{n/2})$ evaluations of f to determine s. This is because on the one hand for any deterministic algorithm that makes a smaller number of evaluations there is a function f such that all values returned by f are distinct, so no information about s is gained; similarly one can show that for any randomized algorithm if f is chosen at random then it is unlikely that the algorithm will gain any information about s in $\ll 2^{n/2}$ evaluations. On the other hand, by making roughly $\Omega(2^{n/2})$ evaluations at random points then by the birthday paradox one will likely obtain $x \neq y$ such that $f(x) = f(y)$, which immediately reveals $s = x + y$.

Simon showed that there is a quantum algorithm that can solve this problem using only $O(n)$ evaluations, provided that the function f can be evaluated "in superposition". The algorithm first evaluates f on a uniform superposition of inputs, as follows:

$$\begin{aligned} |0^n\rangle|0^n\rangle &\mapsto \frac{1}{\sqrt{2^n}} \sum_x |x\rangle|0^n\rangle \\ &\mapsto \frac{1}{\sqrt{2^n}} \sum_x |x\rangle|f(x)\rangle \ . \end{aligned}$$

It then measures the last register in the computational basis, yielding some $y = f(x_0) = f(x_1)$ where x_0 and $x_1 = x_0 + s$ are the two preimages of y under f. The re-normalized post-measurement state is

$$\frac{1}{\sqrt{2}}\big(|x_0\rangle + |x_1\rangle\big)|y\rangle \ . \tag{4.1}$$

Measuring the first register in the Hadamard basis yields a uniformly random $d \in \{0,1\}^n$ such that $d \cdot s = 0$. Repeating the entire procedure $O(n)$ times yields $(n-1)$ linearly independent such d's, which suffices to recover s with high probability.

4.1.2 Instantiating the black box

The main limitation of Simon's problem is that it only provides a *black-box* separation: the quantum advantage holds under the assumption that the classical or quantum algorithms are allowed to evaluate the function f, but they are not given an explicit description of it. Showing that the separation still holds for an explicit choice of the function f is much harder, because it is difficult to rule out some smart behavior for

the classical algorithm that would take advantage of specific code for f; indeed, showing such a separation would be a major breakthrough in quantum algorithms.

This difficulty shouldn't prevent us from toying with the question: Can we identify natural candidates? For example one could take $f(x) = Ax$ for $A \in \mathbb{F}_2^{n\times n}$ a matrix of rank exactly $(n-1)$. In that case the kernel of A is spanned by a single vector $s \in \mathbb{F}_2^n$, and f is exactly 2-to-1: $f(x_0) = f(x_1)$ if and only if $A(x_0 - x_1) = 0$, i.e. $x_0 - x_1$ is either 0 or s. Unfortunately this f is not a good candidate, because there happens to be an efficient classical algorithm that directly solves Simon's problem for it: Gaussian elimination.[2] The example shows that at a minimum we need a function f that is 2-to-1 but such that finding any colliding pair of inputs (x_0, x_1) with $f(x_0) = f(x_1)$ is computationally difficult. In the next section we introduce some background from cryptography that will allow us to make this requirement precise.

4.2 Computational assumptions

So far everything that we have done in the course is "information-theoretic", in the sense that we have not had to fix a model of computation nor discuss efficiency of the various operations that we had our verifiers and provers implement (the only distinction we made is if an operation is classical or if it requires a quantum component). In this lecture we start making assumptions of a computational nature, such as "this problem cannot be solved by this class of adversaries". In this section we briefly review the formalism for making such assumptions precise and apply it to a specific scenario of interest for the lecture.

4.2.1 PPT and QPT procedures

The first thing that we need to make precise is our computational model. Since the protocols we consider involve interaction between a verifier and prover(s) we focus on modeling such devices as machines that perform a computation. Loosely speaking, each device operates in a number of rounds where at each round the device performs a computation that takes it from a certain internal state as well as an input (a message received from another device) to a new internal state and an output (a message that it returns). We will model each such computation as a circuit. A circuit is a sequence of elementary operations called "gates" that operate either on a classical state (in which case the gates can be things like an AND, an OR, a NOT, etc.) or a quantum state (in which case the gates can be things like a 1-qubit Hadamard, a σ_X or σ_Z, a 2-qubit controlled NOT, etc.).[3] To recap, for us a verifier or a prover is specified by a sequence of classical or quantum circuits. We will always assume that the circuits explicitly specify which spaces they are meant to operate on (e.g. verifier's space, message from verifier to prover, etc.).

Next we discuss what it means for a verifier (or prover) to be "efficient". To make this precise we need to talk about *families* of verifiers. We will imagine that there is an underlying size parameter $n \in \mathbb{N}$ (for example, n could be the size of a 3SAT formula that the verifier aims to check, or the number of qubits that she aims to certify) and that the verifier (or prover) is specified by a classical Turing machine M that on input 1^n returns an explicit classical description of a sequence of circuits that can be used to implement the verifier (or prover) for problems of size n. We will say that the verifier (or prover) is *probabilistic polynomial time* (PPT) (resp. *quantum polynomial time* (QPT)) if this Turing machine runs in time polynomial in its input (i.e. polynomial in n; this is why we always assume that n is passed in unary to M) and returns a

[2] In the next lecture we will see that a "noisy" version of f provides a way forward.

[3] To be fully precise we would need to fix a finite gate set for classical circuits and another for quantum circuits. What gate set is used will not matter for us; the only important point is that there exists finite universal gate sets and that all such gate sets are roughly equivalent in terms of how many gates are required to decompose any larger unitary.

family of classical (resp. quantum) circuits. Note that the assumption that the Turing machine is polynomial time immediately implies that the circuits it returns act on polynomially many bits (resp. qubits) and have a polynomial number of classical (resp. quantum) gates.

In a cryptographic context we will generally allow M to take a second input 1^λ for $\lambda \in \mathbb{N}$ called the *security parameter*. While the input size n is governed by the size of the problem, the security parameter can be chosen at will; the larger it is the more "secure" the protocol is supposed to be (for example, the smaller the probability that the verifier makes an incorrect decision or the higher the quality of the certified qubits).

4.2.2 Claw-free functions

Similarly to circuits in the previous section, when we talk about computational *difficulty* of a certain problem we always need to refer to *families* of objects. This is because e.g. for any given function f there is nothing "hard" about the task of recovering specific information about f: if f is fixed everything about it is fixed as well; in particular in the case when f has the structure required for Simon's problem there is a simple algorithm that identifies s, and this is the algorithm that writes s down starting from any initial state.

For this reason we will always consider families of functions $\{f_{pk} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}\}_{pk\in\{0,1\}^{k(\lambda)}}$ where the index $\lambda \in \mathbb{N}$ is called *security parameter* and k and n are polynomially bounded functions of λ; the idea is that for each λ there is a collection of functions, indexed by strings of length $k(\lambda)$ and with the same domain and range, such that the larger the λ the more "complex" the functions are. For example, we could take $k(\lambda) = \lambda^2$, $n(\lambda) = \lambda$, and f_{pk} to be multiplication by the matrix $A \in \{0,1\}^{\lambda\times\lambda}$ obtained by "reshaping" the λ^2-bit string pk into a $\lambda \times \lambda$ square.

Let's give our first definition of a cryptographic property that applies to a family of functions.

Definition 4.1 (Claw-free function family)**.** A family $\mathcal{F} = \{f_{pk} : \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}\}_{pk\in\{0,1\}^{k(\lambda)}}$ is *claw-free* against classical (resp. quantum) adversaries if the following conditions hold:

- f_{pk} can be efficiently evaluated: there is a PPT procedure that given pk and x as inputs returns $f_{pk}(x)$.
- For every $\lambda \in \mathbb{N}$ and $pk \in \{0,1\}^{k(\lambda)}$, f_{pk} is 2-to-1.
- For every PPT (resp. QPT) procedure $\mathcal{A}$ the following holds: (the procedure $\mathcal{A}$ is often personified as the "adversary" trying to demonstrate that the function family is *not* claw-free) There exists a negligible[4] function $\mu : \mathbb{N} \to \mathbb{N}$ such that for every λ,

$$\Pr_{pk\leftarrow_R\{0,1\}^{k(\lambda)}}\left((x_0,x_1)\leftarrow\mathcal{A}(1^\lambda,pk):\ x_0\neq x_1,\ f_{pk}(x_0)=f_{pk}(x_1)\right)\ \leq\ \mu(\lambda)\,.$$

In words, the third condition states that there is no polynomial-time algorithm that given a uniformly chosen index pk for a function from the family is able to return two distinct inputs for the function that constitute a claw.[5]

Remark 4.2. In the definition we require the function family to be parametrized by arbitrary strings pk. In general this requirement can be relaxed; in fact there could even be a single function for every λ. In cryptographic constructions the function family generally comes equipped with a PPT *key generation procedure* GEN that takes 1^λ as input and returns pk.

[4] A function $\mu : \mathbb{N} \to \mathbb{R}$ is called negligible if for every polynomial p, $p(\lambda)\mu(\lambda) \to_{\lambda\to\infty} 0$.

[5] A triple (x_0, x_1, y) such that $x_0 \neq x_1$ and $f(x_0) = f(x_1) = y$ is called a *claw*. To see why, picture the arrows $x_0 \to y$ and $x_1 \to y$ drawn with x_0, x_1 on top of each other on the left and y on the right.

An example of a claw-free family of functions against PPT adversaries can be constructed as follows. (This construction appears in [GMR85], where it is used to construct a digital signature scheme.) Let $N = pq$ be a product of two primes $p \equiv 3 \bmod 8$ and $q \equiv 7 \bmod 8$. This choice ensures that -1 and 2 are not squares $\bmod N$; moreover, if Q_N denotes the set of quadratic residues (i.e. squares) modulo N then $f_0(x) = x^2 \bmod N$ and $f_1(x) = 4x^2 \bmod N$ are both permutations of Q_N. (This fact requires proof but it is a simple exercise in arithmetic.) However, suppose given a claw (x_0, x_1) such that $x_0, x_1 \in Q_N$ and $f_0(x_0) = f_1(x_1)$. Then $x_0^2 = 4x_1^2$ but $x_0 \neq \pm 2x_1 \bmod N$ because $\pm 2x_1 \notin Q_N$. Thus computing the GCD of N with $x_0 \pm 2x_1$ recovers a nontrivial factor.

While this family of functions is claw-free with respect to PPT adversaries, it is clearly not claw-free against QPT adversaries, that can use Shor's algorithm to factor efficiently. We will construct such a function family in the next lecture; for the time being we assume its existence.

4.2.3 Hardcore bits

Following the initial steps of Simon's algorithm as described in Section 4.1.1 when instantiated with any 2-to-1 function enables a quantum device to generate strings $d \in \{0,1\}^n$ such that $d \cdot (x_0 + x_1) = 0$, where (x_0, x_1) are preimages of some $y \in \mathbb{F}_2^n$ by f. Intuitively one might expect that this represents a computational advantage, because d provides an equation in $x_0 + x_1$, which is some information about both preimages together. For example in the case where $x_0 + x_1 = s$, where s is some fixed secret independent of y, we saw that provided the equation d can be assumed to be uniformly distributed among all valid equations in s then running the procedure $O(n)$ times gives sufficiently many equations to recover s.

Unfortunately, as discussed in Section 4.1.2 for the only function that we could think of that has this property it is in fact easy to recover s, even for a classical computer. This suggests that in general the assumption that f satisfies the structure required for Simon's algorithm might be too strong to obtain an explicit candidate. Moreover, recall that at the start of the lecture we pointed out that our goal is not directly to find a task for which there is a quantum computational advantage, but instead we are trying to identify *two* tasks that the quantum device can perform *separately* but not *simultaneously*—if someone, such as a classical device, was able to execute both tasks simultaneously then it would break the computational assumption. What could those two tasks be here?

Starting from the state (4.1) it is natural to measure in the Hadamard basis, obtaining as before an equation d such that $d \cdot (x_0 + x_1) = 0$, but also in the computational basis, obtaining either x_0 or x_1. Given that "honest" measurements in the computational and hadamard basis are incompatible, these are natural candidates for our "qubit." However, we also saw that if $x_0 + x_1 = s$ for some fixed secret s then the Hadamard measurements alone allow us to recover s. So a quantum procedure could recover s "on the side" and then, knowing f explicitly, succeed in any reasonable "test" by using classical operations alone—this would make it very hard for us to identify the "qubit" that the device should have used to recover s (this is similar to the example of Shor's algorithm given at the start of the lecture). But what if the structure of f is a little more complicated, so that e.g. $x_0 + x_1 = g(x_0, s)$ for some function g? In that case a single equation in $g(x_0, s)$ might not be so useful; even many such equations for varying x_0 could be useless since without knowledge of x_0 itself one cannot determine what the equation is about. However, if one was able to obtain x_0 simultaneously with the equation then one would obtain a sequence of (possibly non-linear, depending on g) constraints on s. These considerations motivate the following computational assumption:

Assumption 1 (Adaptive hardcore bit)**.** *There is a claw-free family of functions $\mathcal{F} = \{f_{pk}\}$ such that for*

any QPT adversary $\mathcal{A}$ there is a negligible function μ such that

$$\left|\frac{1}{2} - \Pr_{pk \leftarrow_R \{0,1\}^{k(\lambda)}}\left((x,d) \leftarrow \mathcal{A}(1^\lambda, pk),\ \{x_0,x_1\} \leftarrow f_{pk}^{-1}(f_{pk}(x)) :\ d \neq 0^n \wedge d\cdot(x_0+x_1)=0\right)\right| \leq \mu(\lambda)\,. \tag{4.2}$$

In words, the assumption is that no *quantum* polynomial-time algorithm can *simultaneously* return an element x in the domain of f and an equation d such that, letting $\{x_0,x_1\}$ be the two preimages of $f_{pk}(x)$ under f_{pk} it holds that $d \neq 0^n$ and $d\cdot(x_0+x_1)=0$. Note that although we required $\{f_{pk}\}$ to be claw-free, this requirement is stronger, since any algorithm that can find a claw (x_0,x_1) can be used to break (4.2).

Remark 4.3. Assumption 1 is called *adaptive hardcore bit* for the following reason. Given a function f a hardcore bit for f is a 1-bit function h such that given $f(x)$ (but not x) it is hard to predict $h(x)$. Here, the hardcore bit that underlies the assumption is the function $h(x) = d\cdot(x_0+x_1)$ for any $d \neq 0^n$: the Goldreich-Levin theorem implies that if f is indeed claw-free then it is hard to predict $b(x)$ for a *random* d. The "adaptive" qualifier refers to the fact that in (4.2) we allow the adversary $\mathcal{A}$ itself to select the equation d without requiring that this equation is uniformly distributed (we will see why this is needed in the next section); in particular $\mathcal{A}$ may return always the same d, and this invalidates the classic Golreich-Levin argument. This makes the property harder to satisfy, because more power is given to the adversary. (Note in particular that we had to explicitly require $d \neq 0^n$, as otherwise there is an easy adversary that always succeeds.)

4.3 A computational test for a qubit

We conclude the lecture by giving a "proof of concept" that it is possible to test a qubit based on computational assumptions. Our presentation is a simplified version of the protocol and analysis from [BCM+18]. For a related approach, see [CCKW19].

The main assumption that we make is that there is a function family $\{f_{pk}\}$ that satisfies the hardcore bit assumption, Assumption 1. In fact we will need a little bit more. Let's summarize the requirements as follows:

(**F.1**) There is a claw-free function family $\mathcal{F} = \{f_{pk}\}$ equipped with an efficient key generation procedure $\textsc{Gen}(1^\lambda)$ such that for each key pk the function f_{pk} can be evaluated efficiently.

(**F.2**) The function family $\mathcal{F}$ satisfies the adaptive hardcore bit assumption, Assumption 1.

(**F.3**) $\mathcal{F}$ is equipped with a trapdoor: in addition to pk, $\textsc{Gen}(1^\lambda)$ returns a trapdoor td such that given pk, td and any y in the range of f_{pk} it is possible to efficiently recover the two preimages x_0 and x_1 of y.

(**F.4**) For any pk and any y in the range of f_{pk} the two preimages of y are labelled 'x_0' and 'x_1' using some canonical efficient procedure. That is, given a key pk and an x in the domain of f_{pk} it is possible to efficiently determine if x is the 'x_0' or the 'x_1' preimage of $y = f(x)$. Let $b : \{0,1\}^n \to \{0,1\}$ be this labeling procedure; b may depend on pk.

Let us fix a function family $\mathcal{F}$ satisfying the assumptions (**F.1**) to (**F.4**). We give a protocol based on $\mathcal{F}$. The protocol describes the interaction between a classical polynomial-time verifier and a (possibly quantum) polynomial-time prover. Here, the input to both parties is the security parameter λ; when we refer to PPT or QPT we mean with respect to λ.

1. The verifier generates $(pk, td) \leftarrow \text{GEN}(1^\lambda)$. It sends pk to the prover.
2. The prover returns $y \in \{0,1\}^n$, where $n = n(\lambda)$.
3. The verifier selects a uniformly random challenge $c \leftarrow_R \{0,1\}$ and sends c to the prover.
4. (a) In case $c = 0$ the prover is expected to return an $x \in \{0,1\}^n$. The verifier accepts if and only if $f(x) = 1$.
   (b) In case $c = 1$ the prover is expected to return a $d \in \{0,1\}^n$. The verifier uses td to determine the two preimages (x_0, x_1) of y by f_{pk}. She accepts if and only if $d \cdot (x_0 + x_1) = 0$.

Theorem 4.4. *Let $\mathcal{F}$ satisfy the assumptions (**F.1**) to (**F.4**). Then the following hold:*

- *(Completeness:) There is a QPT prover P which succeeds with probability 1 in the protocol.*
- *(Soundness:) Suppose that a QPT prover P succeeds with probability 1 in the protocol. Then P has a (near-perfect) qubit.*

As we are now accustomed to, we note the slightly informal nature of the theorem and make a few comments:

- First of all, we make explicit the computational assumption: combining Assumption 1 with the requirement that the prover P is QPT effectively means that we are assuming that P does not "have the ability" to violate (4.2). Slightly more formally, in the proof we will show that if P *does not* "have a qubit" then it can be used to construct an adversary $\mathcal{A}$ that violates (4.2). Note also that in the soundness case it should be assumed that P is in fact a family of $\{P_\lambda\}$, one for each possible choice of λ, that can be uniformly generated from λ (i.e. there is a classical Turing machine that takes 1^λ as input and returns a description of a family of circuits that can be used to implement P_λ).
- Second, we ought to be a little more precise as to how P's qubit is specified. The two observables X and Z that define it will be derived from the two measurements that P makes based on the challenges $c = 0$ or $c = 1$. Since these measurements in general have outcomes in $\{0,1\}^n$ some post-processing will be required. Interestingly, the post-processing for the X observable will not be efficient, in the sense that it will require knowledge of td. So, our proof will show that there exists two anti-commuting observables on the Hilbert space of P that can be defined from P's operations *and some classical post-processing*. Since the post-processing is classical we can nevertheless claim in good faith that the "qubit" is located on the prover's space, as we are not injecting any external "quantumness" in it.
- Third, we make the usual comment regarding the assumption that the prover succeeds with probability 1: this assumption is, of course, unrealistic. As in other protocols that we have seen so far the assumption can be lifted at the cost of some amount of work. We will discuss this in more detail when we build on the present protocol to construct a more complex protocol for verifying an entire quantum computation in the next few lectures.
- Finally, an explanation is in order regarding the "(near-perfect)" qualifier. This is an unavoidable consequence of the fact that the protocol relies on a computational assumption. Indeed, consider the following possible behavior for the prover. The prover first devotes a small amount of time to trying their luck at breaking the underlying computational assumption (in our case, the prover could randomly generate candidate trapdoors td' and check if they allow it to invert the function f_{pk}). If the prover succeeds then it can pass in the protocol without manipulating any quantum state, using the fake td' to find a claw that allows it to answer both types of challenges. If it does not succeed then it behaves honestly in the protocol. Such a prover succeeds with probability 1, but the measurement

operators associated with its answers have a part that is "classical" and from which we have no hope of extracting a qubit.

Proof of Theorem 4.4. The completeness part of the theorem is clear. In the first phase the prover proceeds exactly as in Simon's algorithm to obtain the state (4.1). In the second phase, it measures the preimage register in the standard basis in case $c = 0$ and in the Hadamard basis in case $c = 1$, returning the n-bit outcome obtained as its answer. This prover is always accepted with probability 1 in the protocol.

To show the soundness part of the theorem we start with the usual (and, here, crucial) modeling step.

Step 1: Modeling Since we will not need to model the prover's actions in the first phase of the protocol in detail we directly give a name to the state of the prover at the end of step 2; let it be $|\psi\rangle \in \mathcal{H}_{\mathsf{P}}$. This state depends on pk as well as on y; for clarity we suppress this dependence from the notation. Moreover, in general $|\psi\rangle$ may be a mixed state, and we represent it as a pure state for convenience only; in general one could assume that we included a register E to denote an "environment" that holds a purification $|\psi\rangle_{\mathsf{PE}}$ of a general $\rho \in \mathrm{D}(\mathcal{H}_{\mathsf{P}})$.

At the second stage of the protocol the prover is given a challenge $c \in \{0,1\}$ and tasked with responding with an n-bit string, x or d depending on the challenge. In general, x is obtained by performing a POVM $\{\Pi_x\}$ on the prover's entire space, and similarly d is the outcome of a POVM $\{M_d\}$.[6] We make the following observations that allow us to simplify the presentation of these POVM:

- Without loss of generality both Π and M are projective measurements. This is because we can enlarge P and add sufficiently many ancilla qubits initialized to $|0\rangle$ so as to apply Naimark's theorem.
- Without loss of generality, assume that the prover has access to an n-qubit register X initialized to $|0^n\rangle$.
- Without loss of generality, assume that Π is obtained by first applying a unitary transformation U on $\mathcal{H}_{\mathsf{X}} \otimes \mathcal{H}_{\mathsf{P}}$ followed by a standard basis measurement of $\mathcal{H}_{\mathsf{X}} \simeq (\mathbb{C}^2)^{\otimes n}$. Any projective measurement can be put in this form by letting U be any unitary extension of the map

$$|0\rangle|\psi\rangle \in \mathcal{H}_{\mathsf{X}} \otimes \mathcal{H}_{\mathsf{P}} \;\mapsto\; \sum_x |x\rangle\sqrt{\Pi_x}|\psi\rangle \;,$$

  as this map is easily verified to be an isometry on $|0\rangle_{\mathsf{X}} \otimes \mathcal{H}_{\mathsf{P}}$.
- Similarly, without loss of generality assume that M is obtained by first applying a unitary transformation V on $\mathcal{H}_{\mathsf{X}} \otimes \mathcal{H}_{\mathsf{P}}$ followed by a Hadamard basis measurement of $\mathcal{H}_{\mathsf{X}}$.
- Without loss of generality assume that $U = \mathrm{Id}$. This is because we can always redefine the prover's state at the end of step 2 to be $|\psi'\rangle = U|\psi\rangle$, in which case $U' = \mathrm{Id}$ and $V' = VU^\dagger$.

We now introduce observables Z and X on $\mathcal{H}_{\mathsf{X}}$ associated with the prover. For Z, we define it to be

$$Z \;=\; \sum_{x\in\{0,1\}^n} (-1)^{b(x)}|x\rangle\langle x| \;, \tag{4.3}$$

where $b : \{0,1\}^n \to \{0,1\}$ is the function from assumption (**F.4**). Z is efficiently computable since b is. For X, we define it to be

$$X \;=\; \sum_{d\in\{0,1\}^n} (-1)^{d\cdot(x_0+x_1)}\, V^\dagger\big(H_{\mathsf{X}}^{\otimes n} \otimes \mathrm{Id}_{\mathsf{P}}\big)^\dagger\big(|d\rangle\langle d|_{\mathsf{X}} \otimes \mathrm{Id}_{\mathsf{P}}\big)\big(H_{\mathsf{X}}^{\otimes n} \otimes \mathrm{Id}_{\mathsf{P}}\big)V \;, \tag{4.4}$$

[6] We do not need to explicitly mark any dependence of Π or M on pk and y, because without loss of generality the prover has kept a classical copy of these strings in its quantum state $|\psi\rangle$, which can be used as a classical control by both Π and M.

where x_0 and x_1 are the two preimages under f_{pk} of the string y returned by P at step 2. (There is an observable X for each possible string y, but we suppress this dependence from the notation for clarity.) Note that X is *not* efficient, because we are not assuming that determining $x_0 + x_1$ from y is efficient in general. However, X can be computed in a straightforward manner by applying the prover's efficient measurement $\{M_d\}$ followed by (non-efficient) classical post-processing. (We insist on this point to clarify that our definition is not injecting "quantumness" artificially.) Informally, X can be thought of as the observable that determines if the equation d returned by the prover on challenge $c = 1$ is correct or not. In particular, later we will use that for a prover that always succeeds to a challenge $c = 1$ we have $X|\psi\rangle = |\psi\rangle$, i.e. $|\psi\rangle$ is a $+1$ eigenstate of X.

Step 2: Establishing a qubit The goal for the remainder of the proof is to show that $(|\psi\rangle, Z, X)$ form a qubit, i.e. that the two observables X and Z anticommute on $|\psi\rangle$. Informally, this is because if X and Z were jointly measurable then they could be used to simultaneously obtain a preimage of y and a valid equation d in $x_0 + x_1$, thereby violating (**F.2**). We proceed with the details. The heart of the proof is the following claim.

Claim 4.5. *For any $b \in \{0,1\}$,*

$$\big|\langle\psi|Z_b X Z_b|\psi\rangle\big| \;=\; \mathrm{negl}(\lambda)\;,$$

where $Z_b = (\mathrm{Id} + (-1)^b Z)/2$, $\mathrm{negl}(\lambda)$ denotes some negligible function of λ, and the expression on the left should be understood on average over $pk \leftarrow \textsc{Gen}(1^\lambda)$ and the distribution of y as returned by P in the protocol.

Proof. We do the proof for the case $b = 0$, the other case being similar. Suppose for contradiction that there is a polynomial $q : \mathbb{N} \to \mathbb{R}_+$ such that

$$\big|\langle\psi|Z_0 X Z_0|\psi\rangle\big| \;\geq\; \frac{1}{q(\lambda)} \tag{4.5}$$

for infinitely many values of λ. We use this assumption to construct an adversary in (4.2). The adversary proceeds as follows. Given as input 1^λ and pk the adversary first executes the first phase of the prover, obtaining an outcome y and a state $|\psi\rangle$. Then, the adversary measures the n qubits in register X in the computational basis to obtain a value $x \in \{0,1\}^n$. If $b(x) = 0$ then the adversary applies the unitary V and measures register X (again) in the Hadamard basis to obtain $d \in \{0,1\}^n$. The adversary returns the pair (x, d). If $b(x) = 1$ then the adversary chooses $d \in \{0,1\}^n$ uniformly at random and returns d. Since the prover P and b are both efficient, $\mathcal{A}$ is efficient.

Note that this adversary does something "unusual" in the sense that it sequentially applies two operators that the prover would never have applied simultaneously in the protocol. It is to make sense of this sequential application that we made the structural simplifications at the start of the proof. Let's analyze the success probability of $\mathcal{A}$ by using (4.5). There are two cases. Suppose first that the adversary obtains an x such that $b(x) = 0$. Then since P is assumed to succeed with probability 1 in case $c = 0$, we know that necessarily $x = x_0$, and moreover prior to the measurement the support of $|\psi\rangle$ on X contained only the two values $|x_0\rangle$ and $|x_1\rangle$ (as otherwise there would be a chance that the prover returns an invalid preimage to the challenge $c = 0$). Thus by definition of Z in (4.3) the post-measurement state is $Z_0|\psi\rangle$ (suitably re-normalized). The probability that $\mathcal{A}$ obtains $b(x) = 0$ and then a correct equation is then, by definition of X in (4.3) and $X_0 = \frac{1}{2}(\mathrm{Id} + X)$, precisely

$$\langle\psi|Z_0 X_0 Z_0|\psi\rangle \;=\; \frac{1}{2}\big(\langle\psi|Z_0|\psi\rangle + \langle\psi|Z_0 X Z_0|\psi\rangle\big)\;.$$

For the second case assume that $\mathcal{A}$ obtains an x such that $b(x) = 1$. In this case it returns a uniformly random equation; since $x_0 + x_1 \neq 0$ this has probability exactly $\frac{1}{2}$ of being correct. Overall, the adversary's success probability is

$$\frac{1}{2}\langle\psi|Z_1|\psi\rangle + \frac{1}{2}\big(\langle\psi|Z_0|\psi\rangle + \langle\psi|Z_0XZ_0|\psi\rangle\big) \;=\; \frac{1}{2} + \frac{1}{2}\langle\psi|Z_0XZ_0|\psi\rangle \;,$$

where the equality uses $Z_0 + Z_1 = \mathrm{Id}$ to combine the first two terms. Using (4.5), this violates (4.2). □

To conclude the proof of the theorem we need the following simple calculation.

Claim 4.6. *Let X, Z be any two binary observables on $\mathcal{H}$. Then*

$$\frac{1}{4}\{X, Z\}^2 \;=\; XZ_0XZ_0 + Z_1XZ_1X \;.$$

Proof. This can be verified by direct calculation. Using that X and Z are Hermitian and square to identity we get by expanding the square

$$\{X, Z\}^2 = 2 + XZXZ + ZXZX \;. \tag{4.6}$$

Expanding $Z = Z_0 - Z_1$,

$$ZXZ = Z_0XZ_0 + Z_1XZ_1 - Z_0XZ_1 - Z_1XZ_0 \;.$$

Moreover, using $Z_0 + Z_1 = \mathrm{Id}$ it follows that

$$Z_0XZ_0 + Z_1XZ_1 + Z_0XZ_1 + Z_1XZ_0 = X$$

Putting the two equations together, $ZXZ = 2\big(Z_0XZ_0 + Z_1XZ_1\big) - X$. Plugging back into (4.6) and using $X^2 = \mathrm{Id}$ proves the claim. □

Combining Claim 4.6 and Claim 4.5 we make the following calculation:

$$\begin{aligned}\frac{1}{4}\big\|\{X, Z\}|\psi\rangle\big\|^2 &= \langle\psi|XZ_0XZ_0|\psi\rangle + \langle\psi|Z_1XZ_1X|\psi\rangle \\ &= \langle\psi|Z_0XZ_0|\psi\rangle + \langle\psi|Z_1XZ_1|\psi\rangle \\ &= \mathrm{negl}(\lambda) \;,\end{aligned}$$

where to obtain the second line we used that $X|\psi\rangle = |\psi\rangle$ since the prover is assumed to succeed with probability 1 in the protocol (and hence always return a correct equation). This shows that $(|\psi\rangle, Z, X)$ is a near-perfect qubit, completing the proof. □

Bibliography

[AV12] Dorit Aharonov and Umesh Vazirani. Is quantum mechanics falsifiable? A computational perspective on the foundations of quantum mechanics. *arXiv preprint arXiv:1206.3686*, 2012.

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 320–331. IEEE, 2018.

[Bel64] John S Bell. On the einstein podolsky rosen paradox. *Physics Physique Fizika*, 1(3):195, 1964.

[Bre06] Frédéric Brechenmacher. *Histoire du théorème de Jordan de la décomposition matricielle (1870-1930). Formes de représentation et méthodes de décomposition.* PhD thesis, 2006.

[CCKW19] Alexandru Cojocaru, Léo Colisson, Elham Kashefi, and Petros Wallden. Qfactory: classically-instructed remote secret qubits preparation. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 615–645. Springer, 2019.

[CLS17] Richard Cleve, Li Liu, and William Slofstra. Perfect commuting-operator strategies for linear system games. *Journal of Mathematical Physics*, 58(1):012202, 2017.

[CM14] Richard Cleve and Rajat Mittal. Characterization of binary constraint system games. In *International Colloquium on Automata, Languages, and Programming*, pages 320–331. Springer, 2014.

[CR20] Rui Chao and Ben W Reichardt. Quantum dimension test using the uncertainty principle. *arXiv preprint arXiv:2002.12432*, 2020.

[CRSV17] Rui Chao, Ben W Reichardt, Chris Sutherland, and Thomas Vidick. Overlapping qubits. *arXiv preprint arXiv:1701.01062*, 2017.

[CRSV18] Rui Chao, Ben W Reichardt, Chris Sutherland, and Thomas Vidick. Test for a large amount of entanglement, using few measurements. *Quantum*, 2:92, 2018.

[CS17] Andrea Coladangelo and Jalex Stark. Robust self-testing for linear constraint system games. *arXiv preprint arXiv:1709.09267*, 2017.

[EPR35] Albert Einstein, Boris Podolsky, and Nathan Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical review*, 47(10):777, 1935.

[GMR85] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A "paradoxical" solution to the signature problem. In *Advances in Cryptology*, pages 467–467. Springer, 1985.

[GVW01] Oded Goldreich, Salil Vadhan, and Avi Wigderson. On interactive proofs with a laconic prover. In *International Colloquium on Automata, Languages, and Programming*, pages 334–345. Springer, 2001.

[JNV+20] Zhengfeng Ji, Anand Natarajan, Thomas Vidick, John Wright, and Henry Yuen. MIP*= RE. *arXiv preprint arXiv:2001.04383*, 2020.

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

[Mer90] N David Mermin. Simple unified form for the major no-hidden-variables theorems. *Physical review letters*, 65(27):3373, 1990.

[Mer93] N David Mermin. Hidden variables and the two theorems of john bell. *Reviews of Modern Physics*, 65(3):803, 1993.

[MYS12] Matthew McKague, Tzyh Haur Yang, and Valerio Scarani. Robust self-testing of the singlet. *Journal of Physics A: Mathematical and Theoretical*, 45(45):455304, 2012.

[NC02] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[RUV13] Ben W Reichardt, Falk Unger, and Umesh Vazirani. Classical command of quantum systems. *Nature*, 496(7446):456–460, 2013.

[SW87] Stephen J Summers and Reinhard Werner. Maximal violation of bell's inequalities is generic in quantum field theory. *Communications in Mathematical Physics*, 110(2):247–259, 1987.

[VW16] Thomas Vidick and John Watrous. Quantum proofs. *Foundations and Trends® in Theoretical Computer Science*, 11(1-2):1–215, 2016.

[WBMS16] Xingyao Wu, Jean-Daniel Bancal, Matthew McKague, and Valerio Scarani. Device-independent parallel self-testing of two singlets. *Physical Review A*, 93(6):062121, 2016.