Depth-efficient proofs of quantumness

Zhenning Liu[1] and Alexandru Gheorghiu[2]

[1]Department of Physics, ETH Zürich, Switzerland

[2]Institute for Theoretical Studies, ETH Zürich, Switzerland

A proof of quantumness is a type of challenge-response protocol in which a classical verifier can efficiently certify the *quantum advantage* of an untrusted prover. That is, a quantum prover can correctly answer the verifier's challenges and be accepted, while any polynomial-time classical prover will be rejected with high probability, based on plausible computational assumptions. To answer the verifier's challenges, existing proofs of quantumness typically require the quantum prover to perform a combination of polynomial-size quantum circuits and measurements.

In this paper, we give two proof of quantumness constructions in which the prover need only perform *constant-depth quantum circuits* (and measurements) together with log-depth classical computation. Our first construction is a generic compiler that allows us to translate existing proofs of quantumness into constant quantum depth versions. Our second construction is based around the *learning with rounding* problem, and yields circuits with shorter depth and requiring fewer qubits than the generic construction. In addition, the second construction also has some robustness against noise.

Zhenning Liu: zhenliu@ethz.ch

Alexandru Gheorghiu: agheorghiu@ethz.ch

Contents

1 Introduction

Quantum computation is currently in the era of noisy intermediate-scale (NISQ) devices [Pre18]. This means that existing devices have a relatively small number of qubits (on the order of 100), perform operations that are subject to noise and are not able to operate fault-tolerantly. As a result, they are limited to running quantum circuits of small depth in order to obtain high fidelity outputs. Despite these limitations, there have been a number of demonstrations of *quantum computational advantage* [AAB⁺19, ZWD⁺20, WBC⁺21, ZCC⁺22], i.e. performing a task on a quantum device that cannot be *efficiently* reproduced by classical computers, based on plausible complexity-theoretic assumptions [AA11, HM17, BFNV19]. Indeed, with the best known classical algorithms it takes several days of supercomputing power to match the results of the quantum devices, which required only a few minutes to produce [HZN⁺20].

These milestone results illustrate the impressive capabilities of existing quantum devices and highlight the potential of quantum computation in the near future. Yet, one major challenge still remains: how do we know whether the results from the quantum devices are indeed correct? For the existing demonstrations of quantum advantage, verification is achieved using various statistical tests on the output samples from the quantum devices [AAB⁺19, BIS⁺18, ZWD⁺20]. However, performing these tests either involves an exponential-time classical computation or there is no formal guarantee that an efficient classical adversary cannot *spoof* the results of the test [AC17, AG20, PR22].

One conceptually simple way to demonstrate quantum advantage, that's also efficiently verifiable, is to ask the quantum computer to factor large composite integers using Shor's algorithm [Sho94]. Assuming factoring is classically intractable, this task yields a quantum advantage and is tractable to verify (simply multiply the output factors and check if they produce the number to be factored). However, Shor's algorithm requires fault-tolerant quantum computation to perform and so is not suitable for near-term devices [GE21].

An alternative way of performing efficient tests of quantum advantage was initiated by the work of Brakerski et al. in [BCM⁺18]. There, the authors proposed an interactive protocol between a polynomial-time classical *verifier* and a self-claimed polynomial-time quantum *prover*. The verifier issues a number of challenges to the prover and checks the prover's responses, accepting only when the prover answers the challenges correctly. The defining property of such a protocol is that no polynomial-time classical prover can make the verifier accept with high probability, but there exists a quantum polynomial-time strategy that makes the verifier always accept. This is referred to as a *proof of quantumness* protocol. The protocol of Brakerski et al. is based around a family of collision-resistant hash functions known as *trapdoor claw-free functions* (TCFs)[1]. In essence, for the quantum prover to correctly answer the verifier's challenges, one of the things it is required to do is *evaluate these functions in superposition*. With the trapdoor, the verifier is able to check whether the prover performed this evaluation correctly. It can also be shown that for any classical prover to succeed in the protocol, it would effectively have to find collisions for the TCFs. Brakerski et al. showed that TCFs can be constructed assuming the intractability of the *learning with errors* (LWE) problem [Reg09]. In effect, this shows that efficient classical provers cannot succeed in the proof of quantumness, unless LWE is classically tractable. Subsequent works have also shown that TCFs can be based on other problems assumed to be classically intractable, such as factoring, the discrete logarithm problem or ring learning with errors [KMCVY22]. Additionally, TCF-based proofs of quantumness can also be made non-interactive in the random oracle model [BKVV20]. In all of these cases, however, to succeed in the protocol the ideal quantum prover must evaluate the TCFs coherently and this requires, at best, logarithmic quantum depth [GH20].

It is thus the case that, on the one hand, we have statistical tests of quantum advantage that are suitable for NISQ computations but which either require exponential runtime or do not provide formal guarantees of verifiability. On the other hand, we have proofs of quantumness based on plausible computational assumptions, but that are not suitable for NISQ devices, as they require running deep quantum circuits. Is it possible to bridge the gap between the two approaches? One step towards that goal would be to construct proofs of quantumness where the prover is

[1] Concurrently, Mahadev showed how TCFs can be used to perform classical verification of polynomial-time quantum computations [Mah18].

only required to perform *constant-depth* quantum circuits (together with short-depth classical circuits). This would also answer an important theoretical question: can one achieve quantum advantage with constant-depth quantum circuits while also being able to classically verify the results in polynomial time? This is the main result of our work: we give two proof of quantumness constructions in which the prover's evaluation can be performed in constant quantum depth and logarithmic classical depth. For the purposes of certifying quantum advantage, this leads to highly depth-efficient proofs of quantumness. Both constructions also yield depth-efficient protocols for *certifiable randomness generation*, based on the scheme from [BCM$^+$18]. The first construction is a generic compiler that can take existing proof of quantumness protocols, based on TCFs, and convert them into constant-depth versions. The second construction uses a specific TCF based on the *learning with rounding* (LWR) problem [BPR12] and achieves circuits of smaller width and with some amount of noise robustness compared to the generic construction.

1.1 Proofs of quantumness

To explain our approach, we first need to give a more detailed overview of TCF-based proof of quantumness protocols. As the name suggests, the starting point is trapdoor claw-free functions. A TCF, denoted as f, is a type of 2-to-1 one-way function—a function that can be evaluated efficiently (in polynomial time) but which is intractable to invert. The fact that the function is 2-to-1 means that there are exactly two preimages for each image of the function. The function also has an associated trapdoor which, when known, allows for efficiently inverting $f(x)$, for any x. Finally, "claw-free" means that, without knowledge of the trapdoor, it should be intractable to find a pair of preimages, x_0, x_1, such that $f(x_0) = f(x_1)$. Such a pair is known as a *claw*.

For many of the protocols developed so far, an additional property is required known as the *adaptive hardcore bit property*, first introduced in [BCM$^+$18]. Intuitively, this says that for any x_0 it should be computationally intractable to find even a single bit of x_1, whenever $f(x_0) = f(x_1)$. As was shown in [KMCVY22], this property is not required in order to construct proof of quantumness protocols, provided one adds an additional round of interaction in the protocol, as will become clear later. We will refer to TCFs having the adaptive hardcore bit property as *strong TCFs*. More formally, there exists $\lambda_0 > 0$, such that for any $\lambda > \lambda_0$, known as the *security parameter*, a strong TCF, f, is a 2-to-1 function which satisfies the following properties:

1. **Efficient generation.** There is a $poly(\lambda)$-time algorithm that can generate a description of f as well as a trapdoor, $t \in \{0,1\}^{poly(\lambda)}$.

2. **Efficient evaluation.** There is a $poly(\lambda)$-time algorithm for computing $f(x)$, for any $x \in \{0,1\}^\lambda$.

3. **Hard to invert.** Any $poly(\lambda)$-time algorithm has *negligible*[2] probability to invert $y = f(x)$, for x chosen uniformly at random from $\{0,1\}^\lambda$.

4. **Trapdoor.** There is a $poly(\lambda)$-time algorithm that, given the trapdoor t, can invert $y = f(x)$, for any $x \in \{0,1\}^\lambda$.

5. **Claw-free.** Any $poly(\lambda)$-time algorithm has negligible probability to find (y, x_0, x_1), such that $y = f(x_0) = f(x_1)$, $x_0 \neq x_1$.

6. **Adaptive hardcore bit.** Any $poly(\lambda)$-time algorithm succeeds with probability negligibly close to $1/2$ in producing a tuple (y, x_b, d), with $b \in \{0,1\}$, such that

$$y = f(x_0) = f(x_1), \qquad d \cdot (x_0 \oplus x_1) = 0.$$

It should be noted that the properties, as stated here, are not independent of each other. For instance, property 6 implies properties 3 and 5 (and 5 also implies 3). We chose to present the properties this way for the sake of clarity. Without the requirement of an adaptive hardcore bit, we recover the definition of an ordinary or regular TCF. Note that all $poly(\lambda)$-time algorithms mentioned above can be assumed to be classical algorithms.

[2] We say that a function $\mu(\lambda)$ is negligible if for any polynomial $p(\lambda)$, it is the case that $\lim_{\lambda\to\infty} p(\lambda)\mu(\lambda) = 0$.

We now outline the proof of quantumness protocol introduced in [BCM+18]. The classical verifier fixes a security parameter $\lambda > 0$ and generates a strong TCF, f, together with a trapdoor t. It then sends f to the prover. The prover is instructed to create the state

$$\frac{1}{2^{\lambda/2}} \sum_{(b,x)\in\{0,1\}\times\{0,1\}^{\lambda-1}} |b, x\rangle\, |f(b, x)\rangle \tag{1}$$

and measure the second register, obtaining the result y. Note here that the input to the function was partitioned into the bit b and the string x, of length $\lambda - 1$. The string y is sent to the verifier, while the prover keeps the state in the first register,

$$\frac{1}{\sqrt{2}} (|0, x_0\rangle + |1, x_1\rangle)$$

with $f(0, x_0) = f(1, x_1) = y$. The string y essentially *commits* the prover to its leftover quantum state.

The verifier will now instruct the prover to measure this state in either the computational basis, referred to as the *preimage test* or the Hadamard basis, referred to as the *equation test*, and report the result. For the preimage test, the verifier simply checks whether the reported (b, x_b) of the prover satisfies $f(b, x_b) = y$. For the equation test, the prover will report $(b', d) \in \{0, 1\} \times \{0, 1\}^{\lambda-1}$ and the verifier checks whether

$$d \cdot (x_0 \oplus x_1) = b'. \tag{2}$$

In this case, the verifier has to use the trapdoor to recover both x_0 and x_1 from y in order to compute Equation 2.

It is clear that a quantum device can always succeed in this protocol by following the steps outlined above. However, the properties of the strong TCF make it so that no polynomial-time classical algorithm can succeed with high probability. At a high level, the reason for this is the following. Suppose a classical polynomial-time algorithm, $\mathcal{A}$, always succeeds in both the preimage test and the equation test. First, run $\mathcal{A}$ in order to produce the string y. Then, perform the preimage test with $\mathcal{A}$, resulting in (b, x_b), such that $f(b, x_b) = y$. Since $\mathcal{A}$ is a classical algorithm, it can be *rewound* to the point immediately after reporting y and now instructed to perform the equation test. This will result in the tuple (b', d) such that $d \cdot (x_0 \oplus x_1) = b'$. Importantly, $f(0, x_0) = f(1, x_1) = y$. We therefore have an efficient classical algorithm that yields both a hardcore bit for a claw as well as one of the preimages in the claw. As this contradicts the adaptive hardcore bit property, no such algorithm can exist.

As explained in [KMCVY22, BKVV20, ZKML+21], the above argument can be made robust so that the success probabilities of any polynomial-time classical strategy in the two tests satisfy the relation

$$p_{pre} + 2p_{eq} - 2 \leq \text{negl}(\lambda) \tag{3}$$

where p_{pre} denotes the success probability in the preimage test, p_{eq} is the success probability in the equation test and $\text{negl}(\lambda)$ is a negligible function in the security parameter λ.

The protocol described above crucially relies on the adaptive hardcore bit property to achieve *soundness* against classical polynomial-time algorithms. Thus far, this property has only been shown for TCFs constructed from LWE [BCM+18]. It should also be noted that the above protocol is also a scheme for certifiable randomness generation: the bit b obtained in the preimage test can be used as statistical randomness.

Is it possible to construct proof of quantumness protocols based on other computational assumptions than the classical intractability of LWE? Yes, in fact it is not difficult to see that simple proofs of quantumness can be based on the classical intractability of *factoring* or the *discrete logarithm problem* (DLP): ask the prover to solve multiple instances of these problems using Shor's algorithm [Sho94]. Since their solutions can be classically verified efficiently and since the problems are assumed to be classically intractable, this immediately yields a proof of quantumness. The issue with doing this is that the prover has to run large instances of Shor's algorithm, which would require a fault-tolerant quantum computer [GE21]. Instead, as was shown in [KMCVY22], one can construct proofs of quantumness based on factoring or DLP, in which the prover can implement smaller circuits than those required for Shor's algorithm. Such protocols would then be more amenable to experimental implementation on near-term devices.

Let us briefly outline the approach in [KMCVY22]. The idea is to consider TCFs that need not satisfy the adaptive hardcore bit property. Such TCFs can be constructed from more varied computational assumptions than LWE, including factoring, DLP or the ring-LWE problem [LPR10]. All of these are generally considered to be *standard computational assumptions.* Having such a TCF, the protocol then proceeds in the same way as the one outlined above: the verifier requests that the prover prepare the state in 1, measure the function register obtaining the string y and then send it to the verifier. The prover will be left with the state from 1. As before, the verifier will then instruct the prover to perform either a preimage test or an equation test. The preimage test is unchanged: the prover is asked to measure the state from 1 in the computational basis and report back the result.

For the equation test, however, the verifier will first sample a random string $v \in \{0,1\}^\lambda$ and send it to the prover. The prover must then prepare the state

$$\frac{1}{\sqrt{2}}\left(|v \cdot x_0\rangle |x_0\rangle + |v \cdot x_1\rangle |x_1\rangle\right)$$

The x register is measured in the Hadamard basis, resulting in the string $d \in \{0,1\}^{\lambda-1}$ which is sent to the verifier. Upon receiving d, the verifier chooses a random $\phi \in \{\pi/4, -\pi/4\}$ and asks the prover to measure its remaining qubit in the rotated basis

$$\left\{ \begin{array}{c} \cos\frac{\phi}{2}\, |0\rangle + \sin\frac{\phi}{2}\, |1\rangle \\ \cos\frac{\phi}{2}\, |1\rangle - \sin\frac{\phi}{2}\, |0\rangle \end{array} \right\}$$

Denoting as $b \in \{0,1\}$ the prover's response, the verifier uses d and the trapdoor to determine which b is the likely outcome of the measurement and accepts if that matches the prover's response.

The last step in the protocol is reminiscent of the honest quantum strategy in the CHSH game for violating Bell's inequality [CHSH69]. In fact, much like in the CHSH game, the success probability of any classical prover in this protocol is upper bounded by $0.75 + negl(\lambda)$, whereas a quantum prover can succeed with probability $\cos^2(\pi/8) \approx 0.85$. For this reason, the authors of [KMCVY22] refer to the protocol as a *computational Bell test.*

The soundness against classical polynomial-time algorithms follows from a similar rewinding argument to the one outlined for the previous protocol, which used a strong TCF. The main difference is that in this case the verifier introduces an additional challenge for the prover, in the form of the string v and the bit m, from the modified equation test. This equation test is still checking for a hardcore bit of a claw, but unlike the previous protocol, the hardcore bit is no longer adaptive. Intuitively, this is because the verifier chooses which hardcore bit to request; a choice encapsulated by v and m. For more details, we refer the reader to [KMCVY22].

1.2 Our results

In the proofs of quantumness outlined above, the honest quantum prover needs to coherently evaluate a TCF in order to pass the verifier's tests. A first step towards making the protocol depth-efficient would be to make it so that the prover can evaluate the TCF in constant quantum depth. In fact, all that is required is for the prover to prepare the state from 1 in constant depth, since the remaining operations can also be performed in constant depth. To that end, we first give a generic construction allowing the prover to prepare the state in 1, in constant depth, for all existing TCFs. We then consider a second construction with a TCF based on the *learning with rounding* (LWR) problem [BPR12] (a problem that is, for all intents, equivalent to LWE in terms of computational intractability) in which the prover will prepare a state that is essentially equivalent to that in 1. The advantage of this second construction is that the resulting circuits have smaller depth, smaller width (requiring fewer qubits) and have a certain degree of noise robustness, compared to the generic construction. The first construction is presented in detail in Section 3, while the second is in Section 4.

1.2.1 First construction - A generic compiler

We start with the observation from [GH20] that the strong TCFs based on LWE can be evaluated in classical logarithmic depth. In fact this also holds for the TCFs based on factoring, DLP and ring-LWE from [KMCVY22]. As in [GH20], one can then construct *randomized encodings* for these TCFs, which can be evaluated by constant depth classical circuits. A randomized encoding of some function, f, is another function, denoted $\hat{f}$, which is *information-theoretically equivalent* to f. In other words, $f(x)$ can be uniquely and efficiently decoded from $\hat{f}(x, r)$, for any x and for a uniformly random r. In addition, there is an efficient procedure for outputting $\hat{f}(x, r)$, given only $f(x)$. That is to say that $\hat{f}(x, r)$ contains no more information about $f(x)$ than $f(x)$ itself. The formal definition of randomized encodings is given in Subsection 2.4. It was shown in [AIK04] that all functions computable by log-depth circuits admit randomized encodings that can be evaluated in constant depth. However, this doesn't immediately imply that a quantum prover can coherently evaluate these encodings in constant depth. The reason is that these circuits will typically use gates of *unbounded fan-out*. These are gates that can create arbitrarily-many copies of their output. But the gate set one typically considers for quantum computation has only gates of bounded fan-out (single-qubit rotations and the two-qubit $CNOT$, for instance). How then can the prover evaluate the randomized encoding in constant depth with gates of bounded fan-out?

The key observation is that we do not require the prover to be able to evaluate $\hat{f}$ coherently on an arbitrary input, merely on *a uniform superposition over classical inputs*. One of our main results is then the following:

Theorem 1.1 (informal)**.** There is a strategy consisting of alternating constant depth quantum circuits and logarithmic-depth classical circuits for preparing the state:

$$\sum_{x} |x\rangle \, |\hat{f}(x)\rangle \,, \tag{4}$$

up to an isometry, for any $\hat{f}$ that can be evaluated by a constant-depth classical circuit, potentially including unbounded fan-out gates.

To prove this result, we use an idea from the theory of *quantum error-correction*. It is known that cat states (also known as GHZ states) cannot be prepared by a fixed constant-depth quantum circuit [WKST19]. However, if we can interleave short-depth quantum circuits (and measurements) with classical computation, it is possible to prepare cat states in constant quantum-depth. This is akin to performing corrections in quantum error correction, based on the results of syndrome measurements.

In our case, this works as follows. First, prepare a *poor man's cat state* in constant depth, as described in [WKST19]. This is a state of the form

$$X(w) \, \frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}$$

where w is a string in $\{0, 1\}^n$ and

$$X(w) = X^{w_1} \otimes X^{w_2} \otimes ... \otimes X^{w_n},$$

with X denoting the Pauli-X qubit flip operation. As explained in [WKST19], the constant-depth preparation of the poor man's cat state involves a measurement of the parities of neighboring qubits. In other words, the measurement yields the string $z \in \{0, 1\}^{n-1}$, with $z_i = w_i \oplus w_{i+1}$, for $i \in [n - 1]$. Using a log-depth classical circuit, this parity information can be used to determine either w or its binary complement. One then applies the correction operation $X(w)$ to the poor man's cat state, thus yielding the desired cat state

$$\frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}.$$

Having multiple copies of cat states, it is possible to replicate the effect of unbounded fan-out classical gates on a uniform input[3]. To see why, consider the following example. Suppose we have a classical AND gate, having fan-out n. On inputs $a, b \in \{0, 1\}$, it produces the output $c \in \{0, 1\}^n$, with $c_i = a \wedge b$, for all $i \in [n]$. To perform the same operation with bounded fan-out gates, it

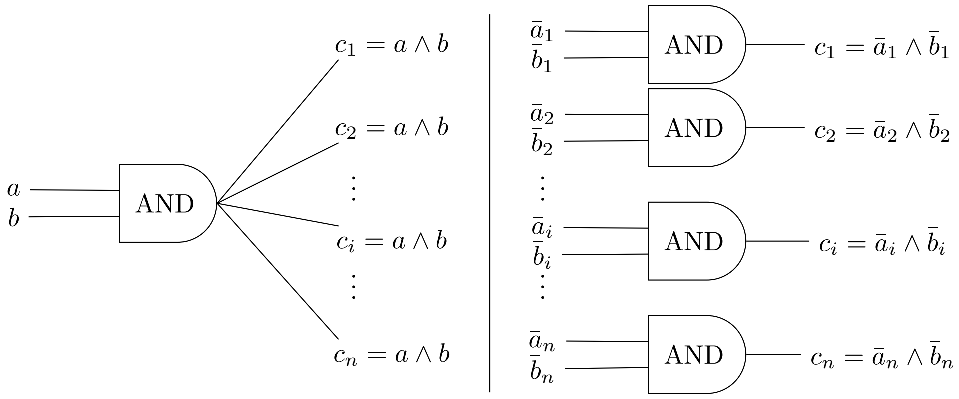

Figure 1: The left-hand side shows an AND gate with fan-out n. The right-hand side is its bounded fan-out equivalent. Here $\bar{a}_i = a$ and $\bar{b}_i = b$. Gates of unbounded fan-out can be implemented with bounded fan-out as long as sufficient copies of the inputs are provided.

suffices to have n copies of a and b. That is, given $\bar{a}, \bar{b} \in \{0,1\}^n$, with $\bar{a}_i = a$, $\bar{b}_i = b$, for all $i \in [n]$, one can compute $c_i = \bar{a}_i \wedge \bar{b}_i$ using n parallel AND gates. This is illustrated in Figure 1.

In our case, each input qubit to the classical function is of the form $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Replacing it with n copies is equivalent to using a cat state $\frac{1}{\sqrt{2}}(|\bar{0}\rangle + |\bar{1}\rangle)$, where $|\bar{0}\rangle = |0\rangle^{\otimes n}$, $|\bar{1}\rangle = |1\rangle^{\otimes n}$. As mentioned, the prover can prepare cat states in constant depth using the "measure-and-correct" trick. It then follows that the prover can also prepare the state

$$\sum_{x} |\bar{x}\rangle |f(x)\rangle \tag{5}$$

where each bit of x is encoded as a cat state having the same number of qubits as the number of input copies required to evaluate f with bounded fan-out gates.

With the ability to prepare the state from 1 (or one equivalent to it, such as the one from 5) in constant quantum depth, the honest prover can then proceed to perform the rest of the steps in the proof of quantumness protocols outlined above. It will measure the image register and report the result to the verifier. The remaining operations can also be performed in constant depth. For the preimage test, the prover simply measures the x register in the computational basis and reports the result. For the equation test, the prover needs to first apply a layer of Hadamard gates to the x register before measuring it in the computational basis. Lastly, for the Bell-type measurement required in the protocol of [KMCVY22], a slightly more involved procedure is used to perform the measurement in constant depth. All of these steps are described in detail in Subsection 3.1.

While we have outlined a procedure for the prover to perform its operations in constant quantum depth, using a randomized encoding of a TCF, it is not immediately clear if we need to also modify the verifier's operations. Indeed, one question that is raised by this approach is whether a randomized encoding of a TCF preserves all the properties of a TCF. If, for instance, the trapdoor property is not preserved, the verifier would be unable to check the prover's responses in the equation test. Our second result resolves this issue:

Theorem 1.2 (informal)**.** A randomized encoding of a (strong) TCF is a (strong) TCF.

This theorem implies that substituting the TCFs used in proofs of quantumness with randomized encodings will not affect the soundness of those protocols. The proof can be found in Subsection 3.2. A similar result was derived in [AIK04], where the authors show that randomized encodings of cryptographic hash functions are also cryptographic hash functions. A (strong) TCF is different, however[4]. To prove this result, first note that most of the TCF properties follow almost

[3] We attribute this idea, of replicating unbounded fan-out with constant-depth quantum circuits and classical measurements, to folklore.

[4] A TCF has exactly two collisions for each image, it has a trapdoor and strong TCFs additionally have the adaptive hardcore bit property. None of these properties are satisfied by generic cryptographic hash functions.

immediately from the definition of a randomized encoding. The more challenging parts concern the existence of a trapdoor and the adaptive hardcore bit property. To show these, we require that the randomized encoding satisfies a property known as *randomness reconstruction* [AIK04]. This states that whenever there is an efficient procedure to invert the original function, f, there should also be an efficient procedure for inverting $\hat{f}$. In particular, this means that given $\hat{f}(x, r)$ it is possible to recover both the input x and the randomness r. In [AIK04], it's mentioned that the randomized encodings used to "compress" functions to constant depth do satisfy the randomness reconstruction property, but no proof is given. We provide a proof in Appendix B.

With the two results of Theorems 1.1 and 1.2, we have that any proof of quantumness using a log-depth computable TCF can be compiled to constant quantum depth for the prover. All of the results for this construction are presented in Section 3, and in Subection 3.3 we give a detailed account of the resources required for the prover to perform this evaluation.

1.2.2 Second construction - Phase encoding and learning with rounding

The second solution to the problem comes from an attempt to directly parallelize the coherent evaluation of the TCF based on LWE, hence to implement the protocol in [BCM+18] in constant quantum depth. We start with the observation that the TCF based on LWE contains only mod-q matrix multiplication and mod-q vector addition operations, where $q \in \mathbb{N}$ is the field size. Since the phases of quantum states have the same periodicity property as the "mod-q" operation, it is natural to consider implementing the mod-q arithmetic with phase Z-rotations (R_z and Controlled-R_z gates). In the standard basis, the R_z operation is expressed as

$$R_z(\theta) = \begin{matrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{matrix} !$$

Note that, for a given cat state, $|\psi\rangle = \frac{1}{\sqrt{2}}(|\bar{0}\rangle + |\bar{1}\rangle)$, applying two R_z phase rotations on *distinct qubits* results in the phases being added into the relative phase of the state. Specifically, if we were to rotate qubit i by θ_i and qubit j by θ_j we would obtain

$$R_z(\theta_i)R_z(\theta_j)\,|\psi\rangle = \frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{i(\theta_i+\theta_j)}\,|\bar{1}\rangle)$$

By taking $\theta_i = \frac{2\pi a}{q}$ and $\theta_j = \frac{2\pi b}{q}$, with $a, b \in \mathbb{Z}_q$, we can see that the net effect is a state with a relative phase proportional to $(a + b) \mod q$,

$$\frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{\frac{2\pi i(a+b)}{q}}\,|\bar{1}\rangle) \tag{6}$$

The key idea is that because these operations commute, *they can be implemented in parallel by acting on distinct qubits*, yielding a constant depth circuit for performing mod-q arithmetic in phase. We denote the state in Equation 6 as $|\phi(a + b)\rangle$[5] and refer to it as a *phase encoding of* $a + b$. Encoding the values of the LWE-based TCF in phase seems to introduce a problem for the protocol. Recall that in the standard proof of quantumness protocol (outlined in Subsection 1.1) the prover encodes evaluations of the function f in the computational basis. If these values were instead encoded in phase, how would the prover be able to obtain an evaluation, y, of the function?

To overcome this obstacle, we consider a different TCF based on a problem known as learning with rounding (LWR) [BPR12, AKPW13]. This problem is equivalent to LWE (for most parameter choices) and was already suggested as a candidate for building TCFs in [BCM+18]. Specifically, denoting now as f an LWR-based TCF, we take

$$f(b, x) : \{0, 1\} \times \mathbb{Z}_q^n \to \mathbb{Z}_p^m = \lfloor \mathbf{A}x + b \cdot (\mathbf{A}s + e) \rceil_p \tag{7}$$

[5] Strictly speaking the notation will refer to states with a relative phase of $\frac{2\pi i(a+b)}{q} - \frac{\pi}{2}$, for reasons that will become clear later. Additionally, when using this notation we will always assume the phases are multiples of the q'th roots of unity as in the example outlined above.

where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $b \in \{0, 1\}$, $x, s \in \mathbb{Z}_q^n$ and $e \in \mathbb{Z}_p^m$ are vectors and $\lfloor \cdot \rceil_p$ denotes rounding over p. By rounding we mean taking the most significant $\log_2 p$ bits of the result[6]. In this case, the result is a vector and the rounding is performed component-wise, so that the output is a vector with entries in $\mathbb{Z}_p$. Note that all matrix multiplications and additions are performed modulo q with $q \gg p$. Intuitively, for small values of e, a typical *claw* of the function should be $(0, x)$ and $(1, x - s)$. This is due to the fact that the rounding operation takes the most significant bits of the output, which are unlikely to be changed when adding a vector e with small entries, component-wise. We refer the reader to the preliminaries in Section 2 for a more detailed explanation of the function and its parameters.

Returning to the idea of the phase encoding, we can now begin to see the reason for choosing this LWR-based function. Consider for the moment the function before rounding,

$$g(b, x) = \mathbf{A}x + b \cdot (\mathbf{A}s + e).$$

Suppose we were to perform a phase encoding of the entries of this function, which we denote as $|\phi(b, x)\rangle$. Now take the i'th entry of that encoding, $|\phi_i(b, x)\rangle$ which encodes the i'th component of $g(b, x)$, denoted $g_i(b, x)$. It is not difficult to see that if we were to measure $|\phi_i(b, x)\rangle$ in the Hadamard basis (or in this case, measure the operator $XX...X$, as we have a rotated cat state), the outcome is most likely to be the *most significant bit* of $g_i(b, x)$. Similarly, if in the phase encoding we used the $q/2$ roots of unity, instead of the q roots of unity, a Hadamard measurement of the encoding would likely yield the second most significant bit. Repeating this $\log_2 p$ times we have a way of probabilistically recovering the output $f_i(b, x) = \lfloor g_i(b, x) \rceil_p$. Of course, due to the probabilistic nature of the measurement, the chance that all bits are recovered correctly will be small. To remedy this issue, we use a classical repetition code. In other words, we view each component of $g(b, x)$ as being repeated several times. When the prover eventually performs its measurements to recover $f(b, x)$ it will take a majority vote for each component. We find that by choosing a suitably large number of repetitions we can make it so that the prover succeeds in evaluating $f(b, x)$ in this way with overwhelming probability.

Our main result is then the following:

Theorem 1.3 (informal)**.** A proof of quantumness protocol, with constant quantum depth and logarithmic depth classical computation, can be constructed based on LWR.

To prove this result, we first need to show that the function f indeed satisfies the properties of a strong TCF. The formal proof of this fact can be found in Subsection 4.1, which is mainly about showing the adaptive hardcore bit property, as all other properties are fairly straightforward.

We next discuss the protocol itself, which is essentially unchanged from that of [BCM$^+$18], except that it uses the LWR-based TCF. Additionally, what changes will be the prover's honest strategy for coherently evaluating this TCF. As mentioned, for this rounding-based function it is possible to coherently evaluate the function in phase, leading to a state that is equivalent (up to an isometry) to

$$\sum_{b,x} |b\rangle_{\mathsf{B}} |x\rangle_{\mathsf{X}} |\phi(b, x)\rangle_{\mathsf{Z}}.$$

To ensure that all mod-q operations, required to prepare this state, can be performed in parallel, the cat states that serve as the basis for the phase encoding must have $\Omega(n \log q)$ qubits. Here, n represents the n rows of the matrix $\mathbf{A}$ and since each component is modulo q, this also contributes a multiplicative $\log q$ factor. As mentioned, we also need to repeat each component in order to guarantee that measurements of the phase-encoded Z register yield a valid image with high probability. We find that the number of repetitions must be $\Omega(n^4 \log^2 n)$ to have a small probability of incorrectly decoding from measurement.

Lastly, we show that the state in the preimage registers, BX, has high overlap with a superposition of preimages, as in the standard version of the protocol. The proof of this fact is based on the observation that while the states $|\phi(b, x)\rangle$ and $|\phi(b', x')\rangle$ are not exactly orthogonal whenever $((b, x), (b', x'))$ does not constitute a claw, they are sufficiently close to orthogonal for *most choices*

[6] In fact this is only true when $q = 2^n$. In our case q will be prime and so the rounding operation, for some value $\alpha \in \mathbb{Z}_q$, is defined as $\lfloor \frac{p}{q} \cdot \alpha \rceil$.

of the matrix $\mathbf{A}$. More specifically, we can show that if $\mathbf{A}$ is uniformly sampled[7] from $\mathbb{Z}_q^{m\times n}$, the overlap between distinct $|\phi(b,x)\rangle$ states decays exponentially in m. On the other hand, if $((b,x),(b',x'))$ does form a claw, we can show that the overlap of $|\phi(b,x)\rangle$ and $|\phi(b',x')\rangle$ is negligibly close to 1. From these facts and the trace-preserving nature of the operations involved, it follows that the state in the preimage register will have high overlap with a superposition of preimages, upon the prover measuring the image register, Z.

An important observation about this construction is that it requires one to perform phase rotations in increments of $\frac{2\pi}{q}$. While such rotation operations are already native to most existing quantum computing architectures, it is also possible to use a constant-size gate set at the expense of making the circuit polynomially wider. This is achieved by approximating the rotation gates to within inverse-polynomial error through the repetition of a fixed set of rotations (see Remark 3.5 in [HŠ05]).

Our second construction is thus an instantiation of the protocol in [BCM$^+$18] with an LWR-based TCF and having the prover perform a phase-encoded evaluation of that function. The main appeal of this construction is that it is much simpler than the generic construction from the previous section and achieves circuits with fewer qubits. Specifically, as computed in Subsections 3.3 and 4.4, for a security parameter $\lambda > 0$, the generic construction uses $O(\lambda^{33})$ qubits, whereas the LWR-based one uses $O(\lambda^8 \log^3 \lambda)$. Additionally, the use of the repetition code and the error-correcting properties of LWR offer the scheme some level of robustness against noise. For the full details and proofs related to this construction, see Section 4.

1.3 Related work

One of the first efficient computational tests of quantum advantage was proposed in [SB09], for certifying that a quantum prover can perform *instantaneous quantum polynomial-time computations* (IQP). However, that test was based on a non-standard hardness assumption and it was later shown that there is an efficient classical algorithm which passes the test [KM19].

The first proof of quantumness based on LWE originated with the work of Brakerski et al. [BCM$^+$18]. This is the proof of quantumness based on a strong TCF outlined in the introduction. As explained there, the protocol also serves as a certifiable random number generator. A subsequent work achieved a non-interactive version of this protocol in the *quantum random-oracle model* [BKVV20]. Notably, in that protocol the adaptive hardcore bit property is not required, however the protocol does make use of a hash function (in addition to the TCF) modeled as a random oracle.

The second proof of quantumness we outlined, based on regular TCFs, was introduced in [KMCVY22]. There the authors achieve more efficient proofs of quantumness by removing the requirement of the adaptive hardcore bit and using TCFs having a lower circuit complexity compared to the ones based on LWE. However, as mentioned, the cost of doing this is introducing additional rounds of interaction between the verifier and the prover (in the form of the Bell-like measurement of the equation test).

In terms of constant quantum depth constructions, it is interesting to contrast our work to that of [CSV21]. There, the authors proposed a protocol for certifiable random-number generation with constant depth quantum circuits. The first difference with respect to our work is that [CSV21] do not base the soundness of their protocol on the classical intractability of some computational problem, such as LWE. Instead, the protocol assumes that the “prover” generating the randomness is a circuit of sub-logarithmic depth (showing that sub-logarithmic classical circuits would not succeed in this task). The second difference is that our protocols require interleaving constant depth quantum circuits with logarithmic depth classical computation, whereas the protocol in [CSV21] only requires the application of a constant depth quantum circuit. Finally, our protocols are interactive, whereas [CSV21] is not.

We also mention the independent work of Hirahara and Le Gall that appeared before ours and which also gives a constant-depth proof of quantumness [HG21]. Similar to our work, they also considered one of the existing proofs of quantumness and made it so that the prover could

[7] Strictly speaking, $\mathbf{A}$ will not be uniform as one needs to sample a matrix $\mathbf{A}$ for which a trapdoor is known, in order to construct an STCF. However, as explained in [BCM$^+$18], the matrix is sampled from a distribution that is statistically close to uniform.

perform its operations in constant quantum depth and using log-depth classical computations. In their case, they use a technique inspired from measurement-based quantum computing to have the prover perform the coherent evaluation of the strong TCF based on LWE. Notably, their prover evaluates that function in the computational basis, unlike our LWR-based scheme which performs the evaluation in phase.

Lastly, we also point out the work of Høyer and Špalek showing that a large class of quantum algorithms can be implemented in constant depth with quantum gates of unbounded fan-out [HŠ05]. In particular, the quantum subroutine of Shor's algorithm can be performed this way. It should then be possible to use the same trick of reproducing unbounded fan-out with bounded fan-out gates, through measurements and classical corrections, as we did for both our constructions. This would then yield a factoring algorithm that uses only constant depth quantum circuits. There are however two downsides to doing this, compared to our approach. First, the resulting algorithm would use classical circuits of supra-logarithmic depth (see also [CW00] for a discussion of this point), in contrast to the logarithmic depth circuits that we obtain [Gal22]. Second, the resulting circuits for factoring would be significantly larger compared to the circuits obtained in our constructions.

1.4 Discussion and open problems

We've shown how existing proof of quantumness protocols can be made to work with a prover that performs constant-depth quantum computations and log-depth classical computations. Thus, all protocols based on TCFs can be compiled to constant-depth versions using randomized encodings and preparations of cat states.

One potential objection to our result is the practicality of this construction. The prover must not only run constant-depth quantum circuits, but it must do so based on the outcomes of previous measurements or based on instructions from the verifier. This is similar to syndrome measurements and corrections in quantum error-correcting codes and so it might seem as if the prover must have the capability of doing fault-tolerant quantum computations. In fact this is not the case. For the protocols based on strong TCFs the number of quantum-classical *interleavings* — that is, the number of alternations between performing a constant depth quantum circuit followed by a log-depth classical circuit — is exactly three. The first is required for the preparation of cat states. In this case, the prover simply needs to apply X corrections conditioned on the outcomes of certain parity measurements. The prover then evaluates the randomized-encoded TCF and measures one of its registers, sending that result to the verifier. Conditioned on its response it either measures the remaining state in the computational basis or in the Hadamard basis. Similar operations are performed for the LWR-based construction. The prover, therefore, needs to do only a very restricted type of conditional operations and is only required to do this three times. Furthermore, the protocol is robust and some degree of noise is acceptable, provided Inequality 3 is violated. When using regular TCFs, in the generic compilation scheme, the protocol requires two additional quantum-classical interleavings, for a total of five. This is due to the Bell-like measurement of that protocol. In both cases, only a small number of quantum-classical interleavings are required, unlike in a fully fault-tolerant computation where many such interleavings would be required [FMMC12].

It would, of course, be desirable to have a single-round proof of quantumness with a constant-depth prover and no quantum-classical interleavings. In other words, a protocol in which the prover has to run a single constant-depth quantum circuit and the verifier is able to efficiently certify that the prover is indeed quantum. Such a result would yield a weak separation between polynomial-time classical computation and constant-depth quantum computation. Basing such a separation on just the classical hardness of LWE seems unlikely[8]. Basing it on the classical intractability of factoring or DLP seems more realistic, as those assumptions already yield a separation between polynomial-time classical computation and logarithmic-depth quantum computation [CW00]. However, it is unclear how to adapt the existing protocols which rely on this commit-and-test approach that requires at least two rounds of interaction. We leave answering this question as an interesting open problem.

Finally, the computational resources required to implement our constant-depth proofs of quantumness are still too high for existing quantum devices. In particular, the resulting quantum

[8] See the first paragraph of the “Our results” subsection in [BKVV20].

circuits can be prohibitively wide to be implemented on existing NISQ devices. However, as we've seen, different implementations can lead to very different qubit requirements. Rough estimates show that our generic construction requires $O(\lambda^{33})$ qubits, while the LWR-based one requires $O(\lambda^8 \log^3 \lambda)$. These substantially different estimates give us some hope that further reducing the qubit requirements is possible. Additional optimizations are likely also possible when considering specific values for the security parameter and the choice of TCF. We therefore also leave as an open problem to reduce the width of these constructions so as to make the protocols better suited for use on near-term devices.

Acknowledgements

AG is supported by Dr. Max Rössler, the Walter Haefner Foundation and the ETH Zürich Foundation.

2 Preliminaries

2.1 Notation and basic concepts

We let $\mathbb{N}$ denote the set of natural numbers, $\mathbb{Z}$ the set of integers, $\mathbb{Z}_q$ the set of integers modulo q, and $\mathbb{R}$ the set of real numbers. The set $\{0,1\}^n$ denotes all binary strings of length n. For some binary string $v \in \{0,1\}^n$, the i'th bit of v is denoted v_i (with $1 \leq i \leq n$). We denote as $|v|$ the *Hamming weight* of v, which is defined as the number of 1's in v, or

$$|v| = \sum_{i=1}^{n} v_i.$$

The *xor* of two bits a, b is $a \oplus b = a + b \; mod \; 2$. This extends to strings so that for $v, w \in \{0,1\}^n$, $v \oplus w$ is their bitwise xor. The *Hamming distance* of the strings v and w is then defined as:

$$d_H(v, w) = |v \oplus w|$$

We will also make use of the bitwise inner product of two strings, defined as:

$$v \cdot w = \sum_{i=1}^{n} v_i \cdot w_i \; mod \; 2.$$

For a bit $b \in \{0,1\}$, we will use $\bar{b}$ to denote a binary string consisting of *copies of* b. That is, $\bar{b} = bbb...b$. The number of copies will generally be clear from the context and will otherwise be specified. We also extend this notation to binary strings. For some string $v \in \{0,1\}^n$, $\bar{v}$ will denote a string in which each bit of v has been repeated. That is, $\bar{v} = v_1v_1...v_1v_2...v_2v_3...v_{n-1}v_n...v_n$.

For any finite set X, we let $x \leftarrow_r X$ denote an element drawn uniformly at random from X. The *total variation distance* between two density functions $f_1, f_2 : X \to [0,1]$ is

$$\mathrm{TVD}(f_1, f_2) = \frac{1}{2} \sum_{x \in X} |f_1(x) - f_2(x)|.$$

For an element $r \in \mathbb{Z}_q$, its unique representative will be $[r]_q \in (-q/2, q/2) \cap \mathbb{Z}$. Following [BCM+18], we use the notation $|r| = |[r]_q|$. For any vector v of n components, its l^2-norm is defined as

$$||v||_2 = \sqrt{\sum_{i=1}^{n} |v_i|^2},$$

and its l^∞ norm is

$$||v||_\infty = \max_i(|v_i|).$$

The *Hellinger distance* between f_1 and f_2 is

$$H^2(f_1, f_2) = 1 - \sum_{x \in X} \sqrt{f_1(x) f_2(x)}.$$

For any discrete probability distribution $p(x)$, its *support* is defined as the set of points where the distribution is positive, $\textsc{Supp}(p(x)) = \{x : \ p(x) > 0\}$.

For a positive $B \in \mathbb{R}$ and positive integer q, the truncated discrete Gaussian distribution over $\mathbb{Z}_q$ with parameter B is supported on $\{x \in \mathbb{Z}_q : \ \|x\| \leq B\}$ and has density

$$D_{\mathbb{Z}_q,B}(x) \ = \ \frac{e^{\frac{-\pi\|x\|^2}{B^2}}}{\sum\limits_{x \in \mathbb{Z}_q, \, \|x\| \leq B} e^{\frac{-\pi\|x\|^2}{B^2}}} \ . \tag{8}$$

We let $negl(x)$ denote a *negligible function.* A function $\mu : \mathbb{N} \to \mathbb{R}$ is negligible if for any positive polynomial $p(x)$ there exists an integer $N > 0$ such that for all $x > N$ it's the case that

$$|\mu(x)| < \frac{1}{p(x)}.$$

We sometimes abbreviate polynomial functions as *poly*. Throughout the paper, λ will denote the *security parameter.* This will be polynomially-related to the input size of all functions we consider. Consequently, all polynomial and negligible functions will scale in λ.

Let $\{D_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{E_\lambda\}_{\lambda \in \mathbb{N}}$ be two families of probability distributions defined on $\{0,1\}^\lambda$. They are *computationally indistinguishable* if for every polynomial-time algorithm $\mathcal{A} : \{0,1\}^\lambda \to \{0,1\}$, it is the case that

$$|\Pr_{x \leftarrow D_\lambda}(\mathcal{A}(x) = 0) - \Pr_{x \leftarrow E_\lambda}(\mathcal{A}(x) = 0)| = negl(\lambda).$$

Letting $g_i \in \mathbb{Z}_q$ with $q \geq 2$, the (mod-q) *phase encoding* of g_i is defined as

$$|\phi_i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{i\phi_i}|1\rangle) \tag{9}$$

where

$$\phi_i = \frac{2\pi g_i}{q} - \frac{\pi}{2}. \tag{10}$$

In terms of quantum information, we follow the usual formalism as outlined, for instance, in [NC02]. All Hilbert spaces are finite dimensional. We use sans-serif font to label spaces that correspond to certain quantum registers. For instance, X will correspond to an n-qubit Hilbert space of inputs to a function. We also extend the bar notation from strings to quantum states. So, for instance $|\bar{0}\rangle = |00...0\rangle$. The multi-qubit *cat state* can then be written as $|\psi\rangle = \frac{1}{\sqrt{2}}(|\bar{0}\rangle + |\bar{1}\rangle)$.

We now recall some standard notions of classical and quantum computation. For more details, we refer the reader to [AB09, NC02].

- The notion of computational efficiency will refer to algorithms or circuits that run in polynomial time.
- We say that an algorithm (or Turing machine) is PPT if it uses randomness and runs in polynomial time. We say it is QPT if it is a quantum algorithm running in polynomial time.
- All Boolean circuits we consider are comprised of AND, OR, XOR and NOT gates.
- We say that a classical gate has *bounded fan-out* if the number of output wires is constant (independent of the length of the input to the circuit). Otherwise, we say it has *unbounded fan-out.*
- For quantum computation we assume the standard circuit formalism with the gate set $\{R_X, R_Y, R_Z, H, CZ, CNOT, CCNOT\}$ and computational basis measurements. Here, R_X, R_Y, R_Z denote rotations along the X, Y and Z axes of the Bloch sphere. More precisely,

$R_W(\theta) = exp(-i\theta W/2)$, with $W \in \{X, Y, Z\}$, the set of Pauli matrices. The allowed rotation angles can be assumed to be multiples of $\pi/4$. In addition, H is the Hadamard operation, CZ is a controlled application of a Pauli-Z gate, $CNOT$ is a controlled application of a Pauli-X gate and $CCNOT$ is a doubly-controlled Pauli-X operation, also known as a Toffoli gate. It should be noted that, apart from $CCNOT$, a number of the existing quantum devices can indeed perform all of these gates natively [AAB+19, AAMA+21, WBD+19].

We say that a computational problem is *intractable* if there is no polynomial-time algorithm solving that problem. Throughout this paper we are only concerned with computational intractability for PPT algorithms. We give a simplified description of some candidate intractable problems of interest:

- **Factoring.** Given a composite integer N, find its prime-factor decomposition. For the specific case of semiprime $N = p \cdot q$, the task is to find primes p and q.
- **Discrete logarithm problem (DLP).** For some abelian group $\mathbb{G}$, given $g \in \mathbb{G}$ and g^k, with $k > 0$, find k.
- **Learning with errors (LWE).** Letting $\mathbb{Z}_q$ be the ring of integers modulo $q \geq 2$, given the matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and the vector $y = \mathbf{A}s + e$, with $s \in \mathbb{Z}_q^n$ and e sampled from a discrete Gaussian distribution over $\mathbb{Z}_q^m$, find s.
- **Ring learning with errors (Ring-LWE).** Letting R_q be a quotient ring $R_q = R/qR$, for some (cyclotomic) ring R over the integers, given $m > 0$ pairs (a_i, y_i) with $a_i \in R_q$ and $y_i = a_i \cdot s + e_i$, $i \leq m$, $s \in R_q$ and each e_i sampled independently from a discrete Gaussian distribution over R_q, find s.

LWE and Ring-LWE are also conjectured to be QPT-intractable [Reg09, LPR10].

2.2 Learning with rounding (LWR)

As learning with rounding is the basis for our second proof of quantumness construction, in this subsection we define the problem and state some of its essential properties, taken from [AKPW13].

Definition 2.1 (Rounding function)**.** For integers $q \geq p \geq 2$, the p-rounding function of an integer α satisfying $0 \leq \alpha < q$ is defined as

$$\lfloor \alpha \rceil_p : \mathbb{Z}_q \rightarrow \mathbb{Z}_p = \quad \frac{p}{q} \cdot \alpha \quad . \tag{11}$$

As mentioned in Subsection 1.2.2, this rounding operation is equivalent to taking the most significant $\log_2 p$ bits of α.

Definition 2.2 (The learning with rounding (LWR) assumption [AKPW13])**.** Suppose $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$, $x \leftarrow_r \mathbb{Z}_q^n$ and $u \leftarrow_r \mathbb{Z}_q^m$, then $(\mathbf{A}, \lfloor \mathbf{A}x \rceil_p)$ and $(\mathbf{A}, \lfloor u \rceil_p)$ are computationally indistinguishable.

Note that this is the *decision* version of LWR. There is also a *search* version, in analogy to LWE. The search version is: given $(\mathbf{A}, \lfloor \mathbf{A}x \rceil_p)$, as above, to find x. Whenever we refer to the "learning with rounding problem" we can use the decision version or the search version interchangeably, as they are equivalent for the parameter choices we use here.

Definition 2.3. (Trapdoor one-way functions from LWR [AKPW13])

1. $\textsc{Gen}(n, m, q)$: an efficient algorithm that receives positive integers n, m, q and samples a matrix $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ and *trapdoor* T with $\mathbf{A}$ being statistically close to uniform.
2. $\textsc{Inv}(T, \mathbf{A}, c)$: an efficient algorithm that receives $T, \mathbf{A}$ in the support of $\textsc{Gen}(n, m, q)$ and $c = \mathbf{A}x + e \in \mathbb{Z}_q^m$ for some $x \in \mathbb{Z}_q^n$ and some error $\|e\|_\infty \leq O \quad \frac{q}{\sqrt{n \log_2 q}}$ and outputs x.
3. $\text{LWR}\textsc{Inv}(T, \mathbf{A}, c)$: for $(\mathbf{A}, T)$ in the support of $\textsc{Gen}(n, m, q)$ and some $c \in \mathbb{Z}_p^m$ such that $c = \lfloor \mathbf{A}x \rceil_p$, the function outputs x efficiently.

Lemma 2.1 (Trapdoors for LWR [AKPW13])**.** There exist efficient Gen and LWRInv functions for any $n \geq 1$, $q \geq 2$, $m \geq O(n \log q)$ and $p \geq O(\sqrt{mn \log q})$. In particular, LWRInv is defined as

$$\text{LWRInv}(T, \mathbf{A}, c) := \text{Inv}(T, \mathbf{A}, \text{Transform}_{q,p}(c)) \tag{12}$$

where

$$\text{Transform}_{q,p}(c) := \frac{q}{p} \cdot c \quad . \tag{13}$$

We also note that for the parameter choices we consider throughout this paper, which are essentially the same as the ones in [BCM+18] (that is, m, n, q, $\|e\|_\infty$ as functions of the security parameter), LWE and LWR are computationally equivalent. In other words, there exists a polynomial-time reduction from LWE to LWR and vice-versa. We refer the reader to [BPR12, AKPW13] for the details.

2.3 Proof of quantumness protocols

2.3.1 Trapdoor claw-free functions

Most proof of quantumness protocols are based on trapdoor claw-free (TCF) functions or noisy trapdoor claw-free functions (NTCF). We start with definition of a TCF, taken from [KMCVY22].

Definition 2.4 (TCF family [KMCVY22])**.** Let λ be a security parameter, K a set of keys, and X_k and Y_k finite sets for each $k \in K$. A family of functions

$$\mathcal{F} = \{f_k : X_k \to Y_k\}_{k \in K}$$

is called a trapdoor claw free (TCF) family if the following conditions hold:

1. **Efficient Function Generation.** There exists a PPT algorithm Gen which generates a key $k \in K$ and the associated trapdoor data t_k:
$$(k, t_k) \leftarrow \text{Gen}(1^\lambda)$$
2. **Trapdoor Injective Pair.** For all keys $k \in K$, the following conditions hold:
   (a) Injective pair: Consider the set R_k of all tuples (x_0, x_1) such that $f_k(x_0) = f_k(x_1)$. Let $X'_k \subseteq X_k$ be the set of values x which appear in the elements of R_k. For all $x \in X'_k$, x appears in exactly one element of R_k; furthermore, $\lim_{\lambda\to\infty} |X'_k|/|X_k| = 1$.
   (b) Trapdoor: There exists a polynomial-time deterministic algorithm $\text{INV}_{\mathcal{F}}$ such that for all $y \in Y_k$ and (x_0, x_1) such that $f_k(x_0) = f_k(x_1) = y$, $\text{INV}_{\mathcal{F}}(t_k, b, y) = x_b$, with $b \in \{0, 1\}$.
3. **Claw-free.** For any non-uniform probabilistic polynomial time (nu-PPT) classical algorithm $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that
$$\Pr\left[f_k(x_0) = f_k(x_1) \wedge x_0 \neq x_1 | (x_0, x_1) \leftarrow \mathcal{A}(k)\right] < \mu(\lambda)$$
where the probability is over both the choice of k and the random coins of $\mathcal{A}$.
4. **Efficient Superposition.** There exists a polynomial-size quantum circuit that on input a key k prepares the state
$$\frac{1}{\sqrt{|X_k|}} \sum_{x \in X_k} |x\rangle |f_k(x)\rangle$$

Next, we define the notion of a noisy TCF, first introduced in [Mah18, BCM+18]. These are TCFs for which the efficient superposition is allowed to be approximate, rather than exact. The outputs of these functions are additionally assumed to be distributions over binary strings, rather than just binary strings. NTCFs, as defined in [BCM+18], also satisfy a property known as the *adaptive hardcore bit* which is independent of the "noisy" aspect of the TCF. As we want to distinguish between TCFs which satisfy this property and those that do not satisfy it, we shall refer to the former as *strong* TCFs and the latter as ordinary TCFs, as per Definition 2.4. Thus, the NTCFs we consider will be referred to as strong NTCFs:

Definition 2.5 (Strong NTCF Family [BCM+18])**.** Let λ be a security parameter. Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets and $\mathcal{D}_{\mathcal{Y}}$ a collection of distributions over $\mathcal{Y}$. Let $\mathcal{K}_{\mathcal{F}}$ be a finite set of keys. A family of functions

$$\mathcal{F} = \left\{ f_{k,b} : \mathcal{X} \to \mathcal{D}_{\mathcal{Y}} \right\}_{k \in \mathcal{K}_{\mathcal{F}}, b \in \{0,1\}}$$

is called a **strong noisy trapdoor claw-free (strong NTCF) family** if the following conditions hold:

1. **Efficient Function Generation.** Same as in Definition 2.4.
2. **Trapdoor Injective Pair.** Same as in Definition 2.4.
3. **Efficient Range Superposition.** For all keys $k \in \mathcal{K}_{\mathcal{F}}$ and $b \in \{0,1\}$ there exists a function $f'_{k,b} : \mathcal{X} \mapsto \mathcal{D}_{\mathcal{Y}}$ such that
   (a) For all $(x_0, x_1) \in \mathcal{R}_k$ and $y \in \textsc{Supp}(f'_{k,b}(x_b))$, $\text{INV}_{\mathcal{F}}(t_k, b, y) = x_b$ and $\text{INV}_{\mathcal{F}}(t_k, b \oplus 1, y) = x_{b \oplus 1}$.
   (b) There exists an efficient deterministic procedure $\text{CHK}_{\mathcal{F}}$ that, on input k, $b \in \{0,1\}$, $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, returns 1 if $y \in \textsc{Supp}(f'_{k,b}(x))$ and 0 otherwise. Note that $\text{CHK}_{\mathcal{F}}$ is not provided the trapdoor t_k.
   (c) For every k and $b \in \{0,1\}$,
   $$\text{E}_{x \leftarrow_U \mathcal{X}} \left[H^2(f_{k,b}(x),\, f'_{k,b}(x)) \right] \leq \mu(\lambda) \,,$$
   for some negligible function $\mu(\cdot)$. Here H^2 is the Hellinger distance. Moreover, there exists an efficient procedure $\text{SAMP}_{\mathcal{F}}$ that on input k and $b \in \{0,1\}$ prepares the state
   $$\frac{1}{\sqrt{|\mathcal{X}|}} \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \sqrt{(f'_{k,b}(x))(y)}\, |x\rangle\, |y\rangle \ .$$
4. **Adaptive Hardcore Bit.** For all keys $k \in \mathcal{K}_{\mathcal{F}}$ the following conditions hold, for some integer w that is a polynomially bounded function of λ.
   (a) For all $b \in \{0,1\}$ and $x \in \mathcal{X}$, there exists a set $G_{k,b,x} \subseteq \{0,1\}^w$ such that $\Pr_{d \leftarrow_U \{0,1\}^w}[d \notin G_{k,b,x}]$ is negligible, and moreover there exists an efficient algorithm that checks for membership in $G_{k,b,x}$ given k, b, x and the trapdoor t_k.
   (b) If
   $$H_k = \left\{ (b, x_b, d, d \cdot (x_0 \oplus x_1)) \,|\, b \in \{0,1\},\ (x_0, x_1) \in \mathcal{R}_k,\ d \in G_{k,0,x_0} \cap G_{k,1,x_1} \right\} \quad (14)$$
   $$\overline{H}_k = \{(b, x_b, d, c) \,|\, (b, x, d, c \oplus 1) \in H_k \} \ , \quad (15)$$
   then for any quantum polynomial-time procedure $\mathcal{A}$ there exists a negligible function $\mu(\cdot)$ such that
   $$\left| \Pr_{(k,t_k) \leftarrow \text{GEN}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in H_k] - \Pr_{(k,t_k) \leftarrow \text{GEN}_{\mathcal{F}}(1^\lambda)} [\mathcal{A}(k) \in \overline{H}_k] \right| \leq \mu(\lambda) \,. \quad (16)$$

As a point of clarification, note that a noisy TCF (NTCF) is a TCF with a modified efficient range superposition property. A strong NTCF is a NTCF with the adaptive hardcore bit property. As mentioned, in [Mah18, BCM+18] NTCFs are not distinguished from strong NTCFs. As an abuse of notation, we will use NTCF and strong NTCF interchangeably.

2.3.2 The BCMVV protocol

The first protocol we mention is the one from [BCM+18], which relies on the adaptive hardcore bit property and so the function family used is NTCF. We outlined the protocol in the introduction, while here we give a step-by-step description of its workings, in Figure 2.
The protocol is complete, in the following sense:

BCMVV protocol

Let $\mathcal{F}$ be an NTCF family of functions. Let λ be a security parameter and $N \geq 1$ a number of rounds. The parties taking part in the protocol are a PPT machine, known as the verifier and a QPT machine, known as the prover. They will repeat the following steps N times:

1. The verifier generates $(k, t_k) \leftarrow \textsc{Gen}(1^\lambda)$. It sends k to the prover.
2. The prover uses k to run $\text{SAMP}_{\mathcal{F}}$ and prepare the state:
$$\sqrt{\frac{1}{|\mathcal{X}|}} \sum_{b\in\{0,1\},x\in\mathcal{X},y\in\mathcal{Y}} \sqrt{(f'_{k,b}(x))(y)}\, |b\rangle_{\mathsf{B}} |x\rangle_{\mathsf{X}} |y\rangle_{\mathsf{Y}} \; .$$
It then measures the Y register, resulting in the string $y \in \{0,1\}^{poly(\lambda)}$ which it sends to the verifier.
3. The verifier selects a uniformly random challenge $c \leftarrow_R \{0,1\}$ and sends c to the prover.
4. (a) (Preimage test:) When $c = 0$, the prover is expected to measure in the standard basis the BX registers of the state leftover in step 2. It obtains the outcomes $b \in \{0,1\}$ and $x \in \{0,1\}^n$, with $n(\lambda) = poly(\lambda)$, which it sends to the verifier. If $\textsc{Chk}_{\mathcal{F}}(k, b, x, y) = 0$ the verifier aborts, otherwise it continues.

   (b) (Equation test:) When $c = 1$, the prover is expected to apply Hadamard gates to each qubit in the BX registers and measure them in the standard basis (equivalently, measure all qubits in the Hadamard basis). It obtains the outcomes $b' \in \{0,1\}$ and $d \in \{0,1\}^n$ which it sends to the verifier. The verifier computes $(x_0, x_1) = \textsc{Inv}_{\mathcal{F}}(t_k, y)$ and rejects if $d \cdot (x_0 \oplus x_1) \neq b'$.

At the end of the N rounds, if the verifier has not aborted it accepts.

Figure 2: The BCMVV proof of quantumness protocol based on NTCFs [BCM+18].

Theorem 2.1 ([BCM+18])**.** A QPT prover, $\mathcal{P}$, following the honest strategy in the BCMVV protocol is accepted with probability $1 - negl(\lambda)$.

The soundness of the protocol against classical provers follows from the following theorem:

Theorem 2.2 ([BCM+18, ZKML+21])**.** For any PPT prover, $\mathcal{P}$, in the BCMVV protocol, it is the case that
$$p_{\text{pre}} + 2p_{\text{eq}} - 2 \leq negl(\lambda) \tag{17}$$
where p_{pre} is $\mathcal{P}$'s success probability in the preimage test and p_{eq} is $\mathcal{P}$'s success probability in the equation test.

Thus, in any run of the protocol, as long as Inequality 17 is violated, we conclude that the prover is quantum.

One known instantiation of the BCMVV protocol, as is described by [BCM+18], is based on the LWE problem. The LWE-based construction is currently the only known instance of a strong NTCF family of functions.

2.3.3 The KMCVY protocol

The BCMVV protocol relies on the adaptive hardcore bit property of NTCFs in order to be sound. However, this property is only known to be true for NTCFs based on LWE. The authors of [KMCVY22] addressed this fact by introducing a proof of quantumness protocol that can use any TCF. As mentioned in the introduction, their protocol is a sort of computational Bell test. We outline it in Figure 3.
The protocol is complete, in the following sense:

Theorem 2.3 ([KMCVY22])**.** A QPT prover, $\mathcal{P}$, following the honest strategy in the KMCVY protocol is accepted with probability $1 - negl(\lambda)$.

The soundness of the protocol against classical provers follows from the following theorem:

KMCVY protocol

Let $\mathcal{F}$ be a TCF family of functions. Let λ be a security parameter, $N \geq 1$ a number of rounds and $T = 1/poly(\lambda)$ a threshold parameter. The parties taking part in the protocol are a PPT machine, known as the verifier and a QPT machine, known as the prover. Before interacting with the prover, the verifier initializes two counters $N_s = 0$, $N_t = 0$. The two will then repeat the following steps N times:

1. The verifier generates $(k, t_k) \leftarrow \textsc{Gen}(1^\lambda)$. It sends k to the prover.
2. The prover uses k to prepare the state:
$$\frac{1}{\sqrt{|X_k|}} \sum_{x \in X_k} |x\rangle_{\mathsf{X}} |f_k(x)\rangle_{\mathsf{Y}}$$
It then measures the Y register, resulting in the string $y \in \{0,1\}^{poly(\lambda)}$ which it sends to the verifier.
3. The verifier selects a uniformly random challenge $c \leftarrow_R \{0,1\}$ and sends c to the prover.
4. (a) (Preimage test:) When $c = 0$, the prover is expected to measure in the standard basis the X register of the state leftover in step 2. It obtains the outcome $x \in \{0,1\}^n$, with $n(\lambda) = poly(\lambda)$, which it sends to the verifier. If $f_k(x) \neq y$ the verifier aborts, otherwise it continues.

   (b) (Computational Bell test:) When $c = 1$,

   i. The verifier sends a random bitstring $v \leftarrow_R \{0,1\}^n$ to the prover.

   ii. The prover creates the state
   $$\frac{1}{\sqrt{2}} \left(|v \cdot x_0\rangle_{\mathsf{A}} |x_0\rangle_{\mathsf{X}} + |v \cdot x_1\rangle_{\mathsf{A}} |x_1\rangle_{\mathsf{X}} \right)$$
   with $f_k(x_0) = f_k(x_1) = y$.

   iii. The prover applies Hadamard gates to all qubits in the X register and measures them in the standard basis. The measurement outcome is denoted $d \in \{0,1\}^n$ and is sent to the verifier.

   iv. The verifier computes $(x_0, x_1) = \textsc{Inv}_{\mathcal{F}}(t_k, y)$. Together with d, the verifier can determine the current state $|\gamma\rangle_{\mathsf{A}} \in \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$ in the prover's A register. It then chooses a random $\phi \in \{\pi/4, -\pi/4\}$ and sends it to the prover.

   v. The prover is expected to measure the qubit in the A register in the basis:
   $$\left\{ \begin{array}{l} \cos\left(\frac{\phi}{2}\right)|0\rangle + \sin\left(\frac{\phi}{2}\right)|1\rangle \\ \cos\left(\frac{\phi}{2}\right)|1\rangle - \sin\left(\frac{\phi}{2}\right)|0\rangle \end{array} \right\}.$$

   vi. The verifier sets $N_s \leftarrow N_s + 1$ if the measurement outcome was the likely one.

If the verifier has not aborted, it will accept if $\frac{N_s}{N_t} - 0.75 \geq T$.

Figure 3: The KMCVY proof of quantumness protocol based on TCFs [KMCVY22].

Theorem 2.4 ([KMCVY22])**.** For any PPT prover, $\mathcal{P}$, in the KMCVY protocol, it is the case that
$$p_{\text{pre}} + 4p_{\text{Bell}} - 2 \leq negl(\lambda) \tag{18}$$
where p_{pre} is $\mathcal{P}$'s success probability in the preimage test and p_{Bell} is $\mathcal{P}$'s success probability in the computational Bell test.

Thus, in any run of the protocol, as long as Inequality 18 is violated, we conclude that the prover is quantum.

In [KMCVY22], the authors provide the following candidate TCFs:

- Rabin's function, or x^2 *mod* n. The TCF properties are based on the computational intractability of factoring.
- A Diffie-Hellman-based function. The TCF properties are based on the computational intractability of DLP.

- A ring-LWE-based function. The TCF properties are based on the computational intractability of ring-LWE.

Of course, the NTCF family based on LWE can also be used.

2.4 Randomized encodings

Randomized encodings (also known as *garbled circuits* [Yao86]) are probabilistic encodings of functions that are information-theoretically equivalent to the functions they encode. The idea of constructing randomized encodings which can be evaluated in constant depth originated with [AIK04]. We restate here the essential definitions and results from that paper.

Definition 2.6 (Randomized encoding [AIK04])**.** Let $f : \{0,1\}^n \to \{0,1\}^l$ be a function and $r \leftarrow_R \{0,1\}^m$ be m bits sampled uniformly at random from $\{0,1\}^m$. We say that a function $\hat{f} : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^s$ is a δ-correct, ϵ-private randomized encoding of f if it satisfies the following properties.

- Efficient generation. There exists a deterministic polynomial-time algorithm that, given a description of the circuit implementing f, outputs a description of a circuit for implementing $\hat{f}$.

- δ-correctness. There exists a deterministic polynomial-time algorithm Dec, called a decoder such that for every input $x \in \{0,1\}^n$, $\Pr_{r \leftarrow_R \{0,1\}^m}[\text{Dec}(\hat{f}(x,r)) \neq f(x)] \leq \delta$.

- ϵ-privacy. There exists a PPT algorithm S, called a simulator, such that for every $x \in \{0,1\}^n$, $\text{TVD}(S(f(x)), \hat{f}(x,r)) \leq \epsilon$.

A *perfect randomized encoding* is one for which $\delta = 0$ (perfect correctness) and $\epsilon = 0$ (perfect privacy). Note that for perfect encodings $f(x)$ can always be reconstructed from $\hat{f}(x,r)$. Additionally, perfect privacy means that $\hat{f}(x,r)$ encodes as much information about x as $f(x)$. An important property of perfect encodings that we will use is that of *unique randomness*:

Theorem 2.5 (Unique randomness [AIK04])**.** Suppose $\hat{f}$ is a perfect randomized encoding of f. Then for any input x, the function $\hat{f}(x,\cdot)$ is injective; namely, there are no distinct r,r' such that $\hat{f}(x,r) = \hat{f}(x,r')$. Moreover, if f is a permutation, then so is $\hat{f}$.

The main result in [AIK04] is the following:

Theorem 2.6 ([AIK04])**.** Any Boolean function that can be computed by a log-depth circuit, admits a perfect randomized encoding that can be computed in constant depth.

In fact a more general result is shown in [AIK04], however the result of the above theorem is sufficient for our purposes. We also require the following result:

Lemma 2.2 (Randomness reconstruction)**.** Given x and $\hat{f}(x,r)$, where $\hat{f}$ is a randomized encoding following the construction from [AIK04], there is a deterministic polynomial-time algorithm, denoted Rrc, for computing the randomness r.

Note that this property is not universal to randomized encodings, in that it cannot be derived from the definition of randomized encodings. However, the property is satisfied by the specific encodings defined in [AIK04]. This fact is mentioned in [AIK04], however no formal proof is provided. We outline their construction in Appendix A and prove the randomness reconstruction property in Appendix B.

Finally, we show the following fact concerning randomized encodings of functions that may have collisions:

Lemma 2.3 (Collision preservation)**.** For every x_1, x_2 with $x_1 \neq x_2$ for which $f(x_1) = f(x_2)$ there exist unique r_1 and r_2 such that $\hat{f}(x_1, r_1) = \hat{f}(x_2, r_2)$. In addition, for every $(x_1, r_1), (x_2, r_2)$, $x_1 \neq x_2$, such that $\hat{f}(x_1, r_1) = \hat{f}(x_2, r_2)$ it is the case that $f(x_1) = f(x_2)$.

Proof. Perfect privacy says that there exists a polynomial-time simulator S, such that for all x, it should be that $\text{TVD}(S(f(x)), \hat{f}(x,r)) = 0$, where TVD is the total variation distance and r is sampled uniformly at random. Essentially, S should always be able to sample from the set of randomized encoding values that can be decoded to $f(x)$ (i.e. all $\hat{f}(x,r)$, for all r).

But now suppose we have x_1 and x_2 such that $f(x_1) = f(x_2)$. By perfect privacy it must be that $\text{TVD}(S(f(x_1)), \hat{f}(x_1,r_1)) = 0$ and $\text{TVD}(S(f(x_2)), \hat{f}(x_2,r_2)) = 0$, for uniform r_1 and r_2. Since $f(x_1) = f(x_2)$, it must be that $\text{TVD}(\hat{f}(x_1,r_1), \hat{f}(x_2,r_2)) = 0$. In other words, $\hat{f}(x_1,r_1)$ and $\hat{f}(x_2,r_2)$ are the same distribution (for random choices of r_1 and r_2) and so the randomized encodings that can be decoded to $f(x_1) = f(x_2)$ are the same for both x_1 and x_2.

Moreover, unique randomness (Theorem 2.5) ensures that there are no distinct r_1 and r_1' such that $\hat{f}(x_1,r_1) = \hat{f}(x_1,r_1')$ (with the analogous statement holding for the x_2 case). Thus, for uniform r_1, $\hat{f}(x_1,r_1)$ is the uniform distribution over all randomized encodings which decode to $f(x_1) = f(x_2)$. As $\hat{f}(x_2,r_2)$ is the same distribution (for uniform r_2), it is the case that there are unique r_1 and r_2 such that $\hat{f}(x_1,r_1) = \hat{f}(x_2,r_2)$. This shows the first part of the lemma, that for every x_1, x_2 with $x_1 \neq x_2$ for which $f(x_1) = f(x_2)$ there exist unique r_1 and r_2 such that $\hat{f}(x_1,r_1) = \hat{f}(x_2,r_2)$.

Next, consider $(x_1,r_1),(x_2,r_2)$, $x_1 \neq x_2$, such that $\hat{f}(x_1,r_1) = \hat{f}(x_2,r_2)$. Since $\textsc{Dec}(\hat{f}(x_1,r_1)) = f(x_1)$ and $\textsc{Dec}(\hat{f}(x_2,r_2)) = f(x_2)$, because $\hat{f}(x_1,r_1) = \hat{f}(x_2,r_2)$ it follows that $\textsc{Dec}(\hat{f}(x_1,r_1)) = \textsc{Dec}(\hat{f}(x_2,r_2))$ and so $f(x_1) = f(x_2)$.

Hence, the collisions of the original function are exactly preserved by the encoding. □

3 Generic proofs of quantumness in constant quantum depth

We now have all the tools for presenting our generic compiler which can take the two proof of quantumness protocols from Subsection 2.3 and map them to equivalent protocols in which the prover's operations require only constant quantum depth and logarithmic classical depth. The idea is the following: provided the (N)TCF of the original protocol can be evaluated in log depth, simply *replace it with a constant-depth randomized encoding*, as follows from Theorem 2.6. In other words, $y = f(x)$ should be replaced by $\hat{y} = \hat{f}(\hat{x})$ where $\hat{x} = (x,r)$ and r denotes the randomness of the encoding. As mentioned, it was shown in [GH20, KMCVY22] that the (N)TCFs of the two proofs of quantumness considered here, can indeed be performed in classical logarithmic depth. Thus, to show that our construction works, we prove two things:

1. The prover can evaluate $\hat{f}$ coherently in constant quantum depth (as well as perform its remaining operations in constant depth). This is the completeness condition of the protocol shown in Subsection 3.1.

2. A randomized encoding of a (N)TCF is itself a (N)TCF. This means that the modified protocol is sound against classical polynomial-time provers. We show this in Subsection 3.2.

3.1 Completeness

To show completeness, we give a strategy for an honest prover, that interleaves constant-depth quantum circuits and log-depth classical circuits, to succeed in the proofs of quantumness described in Section 3. We assume that the (N)TCFs used in those protocols can be evaluated in constant classical depth and denote the corresponding function as $\hat{f}_k$. These circuits are allowed to contain gates of unbounded fan-out. We can always map such a circuit to one that uses only gates of bounded fan-out, provided multiple copies of the input bits are provided. The intuition for this was mentioned in the Introduction and in Figure 1. We will assume each input bit of the initial circuit has been copied k times.

The first step is preparing the state corresponding to a coherent evaluation of the (N)TCF over a uniform superposition of inputs:

$$|\psi\rangle = \sum_{b\in\{0,1\}} \sum_{\hat{x}\in\{0,1\}^{poly(\lambda)}} |b\rangle_{\mathsf{B}} \, |x\rangle_{\mathsf{X}} \, |\hat{f}_k(b,x)\rangle_{\mathsf{Y}} \,, \tag{19}$$

where the B and X registers store the inputs of $\hat{f}$ and the Y register will store the computed value of $\hat{f}$. As a slight abuse of notation, we omit the normalization term and assume the state is an equal superposition.

Instead of preparing the state in Equation 19, we will prepare a state that is essentially equivalent to it, namely:

$$|\psi\rangle = \sum_{b\in\{0,1\}} \sum_{\hat{x}\in\{0,1\}^{poly(\lambda)}} |\bar{b}\rangle_{\mathsf{B}} |\bar{x}\rangle_{\mathsf{X}} |\hat{f}_k(b,x)\rangle_{\mathsf{Y}} , \tag{20}$$

where $|\bar{b}\rangle = |b\rangle^{\otimes k}$ and $|\bar{x}\rangle = |x\rangle^{\otimes k}$. We view the X register as consisting of multiple sub-registers, one for each bit in x. In other words[9], if $x = x_1 x_2 ... x_n$, with $n(\lambda) = poly(\lambda)$, and $\bar{x} = \bar{x}_1 \bar{x}_2 ... \bar{x}_n$, we assume $\mathsf{X} = \mathsf{X}_1 \otimes \mathsf{X}_2 \otimes ... \otimes \mathsf{X}_n$. Here, X_i holds the state $\sum_{x_i\in\{0,1\}} |\bar{x}_i\rangle$.

The prover starts by preparing:

$$|\psi_0\rangle = \sum_{b,\hat{x}} |\bar{b}\rangle_{\mathsf{B}} |\bar{x}\rangle_{\mathsf{X}} |0\rangle_{\mathsf{Y}} . \tag{21}$$

Note that the B and X registers contain cat states. These can be prepared in constant quantum depth, together with logarithmic classical depth. As outlined in the introduction, the idea is to first prepare a poor man's cat state in constant depth, as described in [WKST19]. The prover then uses the parity information from the prepared poor man's cat state to perform a correction operation consisting of Pauli-X gates. Determining where to perform the X gates from the parity information requires logarithmic classical depth. The X corrections will map the poor man's cat states to cat states.

Next, the function $\hat{f}$ needs to be evaluated and the outcome will be stored in Y register. With multiple copies of the input, the circuit evaluating $\hat{f}$ consists only of gates with bounded fan-out. It can therefore be mapped to an equivalent constant depth quantum circuit (having twice the depth, so as to perform the operations reversibly) consisting of Toffoli, Pauli-X and $CNOT$ gates. Evaluating this circuit on the state from 21 will result in the state from 20, as intended.

The prover is then required to measure the Y register and report the outcome to the verifier. This adds one more layer to the circuit. The measured state will collapse to

$$|\psi_y\rangle = \sum_{b\in\{0,1\}} |\bar{b}\rangle_{\mathsf{B}} |\bar{x}_b\rangle_{\mathsf{X}} |y\rangle_{\mathsf{Y}} .$$

In the preimage test, the prover will also measure this state in the computational basis and report the outcome to the verifier.

The next steps will differ for the two protocols.

1. **For the BCMVV protocol:** In the equation test, the prover applies a layer of Hadamard gates on the qubits in B and X. It then measures them in the computational basis, denoting the results as $b' \in \{0,1\}^k$ and $d \in \{0,1\}^{n\cdot k}$. In the original protocol, b' was one bit and d was n bits and they satisfy the relation $d \cdot (x_0 \oplus x_1) = b'$. To arrive at that result, the prover will xor all the bits in b' and all bits in each k-bit block of d and report those results to the verifier. Note that the distributions of these xor-ed outcomes is the same as the distribution over the outcomes of a Hadamard-basis measurement of:

$$\sum_{b\in\{0,1\}} |b\rangle_{\mathsf{B}} |x_b\rangle_{\mathsf{X}} .$$

2. **For the KMCVY protocol:** In the computational Bell test, the prover receives the string v from the verifier. The original protocol has the prover use an ancilla qubit to store the bitwise inner product $v \cdot x_b$. However, such a multiplication requires *serial* $CNOT$ gates which cannot be performed in constant depth. We therefore use a multi-qubit ancila register initalized as a cat state $|a\rangle_{\mathsf{A}} = \frac{|0\rangle^{\otimes n} + |1\rangle^{\otimes n}}{\sqrt{2}}$. For every bit v_i, in v, if $v_i = 1$, the prover applies

[9] Note that this is the only place where a subscript on x is used to denote a bit of x. Throughout the rest of the section, x_b will denote a specific x *string*, and *does not* refer to the b'th bit of the string x.

a controlled-Z (CZ) gate with control qubit any of the qubits in X_i and target qubit $|a\rangle_i$. The resulting state will be

$$\sum_{b\in\{0,1\}} \frac{|0\rangle_{\mathsf{A}}^{\otimes n} + (-1)^{v\cdot x_b} |1\rangle_{\mathsf{A}}^{\otimes n}}{\sqrt{2}} |\bar{x}_b\rangle_{\mathsf{X}} = \sum_{b\in\{0,1\}} |(-1)^{v\cdot x_b}\rangle_{\mathsf{A}} |\bar{x}_b\rangle_{\mathsf{X}}$$

where we denote $|(-1)^{v\cdot x_b}\rangle = \sum_{b\in\{0,1\}} \frac{|0\rangle_{\mathsf{A}}^{\otimes n} + (-1)^{v\cdot x_b}|1\rangle_{\mathsf{A}}^{\otimes n}}{\sqrt{2}}$. Next, the prover is required to measure X in the Hadamard basis yielding the result $d \in \{0,1\}^{n\cdot k}$. Once again, in the original protocol d is an n-bit string. As in the BCMVV protocol, this is "fixed" by having the prover xor each k-bit block of d and report those outcomes to the verifier. The verifier can then use this result to determine the state in the ancilla register.

After the measurement, the ancilla register will be in the state $|\gamma\rangle_{\mathsf{A}} \in \{|\bar{0}\rangle, |\bar{1}\rangle, |\bar{+}\rangle, |\bar{-}\rangle\}$ where $|\bar{\pm}\rangle = \frac{|\bar{0}\rangle \pm |\bar{1}\rangle}{\sqrt{2}}$[10]. As the last step, the prover receives $\phi \in \{-\pi/4, \pi/4\}$. The original protocol requires him to measure the ancilla register in the rotated basis

$$\cos(\phi/2)\,|\bar{0}\rangle + \sin(\phi/2)\,|\bar{1}\rangle$$
$$\cos(\phi/2)\,|\bar{1}\rangle - \sin(\phi/2)\,|\bar{0}\rangle$$

and report the result, b'. But how does the prover perform this measurement in constant depth? We give an approach that requires one more round of interleaving constant-depth quantum circuits and a log-depth classical computation. The basic idea is to reduce the multi-qubit state in the ancilla to a single-qubit state, i.e. $\{|\bar{0}\rangle, |\bar{1}\rangle, |\bar{+}\rangle, |\bar{-}\rangle\} \to \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$. This reduction needs to be done in such a way that $\{|\bar{0}\rangle, |\bar{1}\rangle\} \to \{|0\rangle, |1\rangle\}$ and $\{|\bar{+}\rangle, |\bar{-}\rangle\} \to \{|+\rangle, |-\rangle\}$. Once this is done, the resulting qubit can be measured in the rotated basis.

To perform the reduction, the prover first measures all but one qubit of $|\gamma\rangle_{\mathsf{A}}$ in the Hadamard basis. Denote this $(n-1)$-bit outcome as w. If the initial state was $|\bar{0}\rangle$ or $|\bar{1}\rangle$, the unmeasured qubit will be $|0\rangle$ or $|1\rangle$ respectively. If the initial state was $|\bar{\pm}\rangle$, it can be re-expressed as

$$\begin{aligned}
|\bar{\pm}\rangle &\propto |0\rangle\,|00...0\rangle \pm |1\rangle\,|11...1\rangle \\
&\propto |0\rangle\,(|+\rangle + |-\rangle)^{\otimes n-1} \pm |1\rangle\,(|+\rangle - |-\rangle)^{\otimes n-1} \\
&\propto \sum \left(|0\rangle \pm (-1)^{|w|}\,|1\rangle\right)|w\rangle \\
&\propto \sum_{w} Z^{|w|\ mod\ 2}\,|\pm\rangle\,|w\rangle
\end{aligned}$$

Thus, the qubit after the measurement will be $Z^{|w|\ mod\ 2}\,|\pm\rangle$. The prover will apply the $Z^{|w|\ mod\ 2}$ operation to this qubit. In this way, the state $|\bar{\pm}\rangle$ is reduced to $|\pm\rangle$.

Finally, the prover has to measure the qubit in the rotated basis and report the outcome. This can be done in constant depth by rotating the qubit appropriately and measuring in the standard basis. As in the original protocol, this prover will pass the verifier's checks with probability $\cos(\pi/8)^2 \approx 85\%$.

3.2 Soundness

We do not need to prove soundness from scratch for our modified protocols. Instead, since our only change was to replace the (N)TCFs used in the protocols with randomized encodings, we will have the same soundness as the original constructions provided randomized encodings of (N)TCFs are still (N)TCFs. That is what we show here.

Theorem 3.1. A perfect randomized encoding of a (N)TCF, satisfying the randomness reconstruction property, is still a (N)TCF.

[10] Note that here the bar notation, $|\bar{a}\rangle$, refers to an n-fold repetition, rather than a k-fold one as in the previous case. That is, here $|\bar{a}\rangle = |a\rangle^{\otimes n}$.

Proof. We show this result for NTCFs specifically, since the TCF case is subsumed. The idea of the proof is to show that every property of a NTCF is also satisfied by its randomized encoding.

1. **Efficient Function Generation.** By definition, randomized encodings can be efficiently generated given a description of the function to be encoded. In this case, the description is given by the public key produced by the PPT algorithm $\mathrm{GEN}_{\mathcal{F}}$. More precisely, $\mathrm{GEN}_{\mathcal{F}}$ generates the key $k \in \mathcal{K}_{\mathcal{F}}$ together with a trapdoor t_k. The generating procedure for the encoding will run $\mathrm{GEN}_{\mathcal{F}}$ and output k, the efficient circuit for generating a randomized encoding and the trapdoor t_k. Schematically,

$$(\hat{f}_{k,b}, t_k) \xleftarrow{\text{randomized encoding}} (f_{k,b}, t_k) \equiv (k, t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda) \ .$$

2. **Trapdoor Injective Pair.**

   (a) *Trapdoor*: Due to perfect correctness, $\textsc{Supp}(\hat{f}_{k,b}(x_0, r_0)) \cap \textsc{Supp}(\hat{f}_{k,b}(x_1, r_1)) = \emptyset$ is satisfied since if $\textsc{Supp}(\hat{f}_{k,b}(x_0, r_0)) \cap \textsc{Supp}(\hat{f}_{k,b}(x_1, r_1)) \neq \emptyset$, then perfect correctness leads to $\textsc{Supp}(f_{k,b}(x_0)) \cap \textsc{Supp}(f_{k,b}(x_1)) \neq \emptyset$ which violates the trapdoor injective pair property of the original function f. The efficient deterministic algorithm for inverting the randomized encoding also exists and is defined as $\textsc{Inv}_{\hat{\mathcal{F}}}(t_k, b, \hat{y}) = \textsc{Rrc} \circ \textsc{Inv}_{\mathcal{F}} \circ \textsc{Dec}(t_k, b, \hat{y})$, i.e. the composition of the decoding operation for the encoding, the original $\textsc{Inv}_{\mathcal{F}}$ procedure of the NTCF and the randomness reconstruction procedure (see Lemma 2.2).

   (b) *Injective pair*: Let $\hat{R}_k$ be the set of all tuples of the form $((x_0, r_0), (x_1, r_1))$ such that $\hat{f}_{k,0}(x_0, r_0) = \hat{f}_{k,1}(x_1, r_1)$. Additionally, let $\hat{X}'_k \subseteq \hat{X}_k$ be the set of values (x, r) which appear in the elements of $\hat{R}_k$. It is the case that every $(x, r) \in \hat{X}'_k$ appears in exactly one element of $\hat{R}_k$. This is because, using the collision-preservation property (Lemma 2.3), it must be that $\hat{f}_{k,0}(x_0, r_0) = \hat{f}_{k,1}(x_1, r_1)$ only if $f_{k,0}(x_0) = f_{k,1}(x_1)$ and only for unique r_1 and r_2. We also know from the injective pair property of $f_{k,b}$, that every x appears in exactly one tuple defining a collision for $f_{k,b}$.

   Also note that $|\hat{X}_k| = 2^m |X_k|$, where $|r| = m$. In other words, the set of possible inputs for $\hat{f}_{k,b}$ is 2^m times larger than that of $f_{k,b}$, as for every input, x, we also have the m-bit string r. The collision preservation property (Lemma 2.3) also ensures that $|\hat{X}'_k| = 2^m |X'_k|$. Since we know that $\lim_{\lambda \to \infty} |X'_k|/|X_k| = 1$ it also follows that $\lim_{\lambda \to \infty} |\hat{X}'_k|/|\hat{X}_k| = 1$.

3. **Efficient Range Superposition.** The efficient range superposition property of the original function f means there's an efficient quantum procedure to create a state approximating a superposition over the range of f. Assume we add an additional register, $\mathcal{R}$, to represent the randomness of the encoding, $\hat{f}$, and initialize it as a uniform superposition over computational basis states. We can now combine the efficient procedure for generating $\hat{f}$ with the procedure for generating the range superposition of f and apply them coherently on $\mathcal{R}$. This will then yield the desired state

$$\sum_{x,r,y} \sqrt{(\hat{f}'_{k,b}(x, r))(y)} \, |x\rangle \, |r\rangle \, |y\rangle \ ,$$

   suitably normalized.

4. **Adaptive Hardcore Bit.** We prove this property by contradiction. Assume there exists a QPT adversary $\hat{\mathcal{A}}$ that breaks the adaptive hardcore bit property for the randomized encoding. This means that there exists a non-negligible function $p(\lambda)$ that satisfies

$$\Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)} [\hat{\mathcal{A}}(k) \in \hat{H}_k] - \Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)} [\hat{\mathcal{A}}(k) \in \overline{\hat{H}}_k] \ \geq p(\lambda)$$

   where

$$\hat{H}_k = \ (b, \hat{x}_b, \hat{d}, \hat{d} \cdot (\hat{x}_0 \oplus \hat{x}_1)) \mid b \in \{0, 1\}, \ (\hat{x}_0, \hat{x}_1) \in \hat{\mathcal{R}}_k, \ \hat{d} \in \hat{G}_{k,0,x_0} \cap \hat{G}_{k,1,x_1}$$

and

$$\overline{\hat{H}}_k = \{(b, \hat{x}_b, \hat{d}, c) \mid (b, \hat{x}, \hat{d}, c \oplus 1) \in \hat{H}_k \quad .$$

By definition $\hat{x}_b = (x_b, r_b)$, therefore $\hat{d}$ can be split into (d_x, d_r) such that

$$\hat{x}_b \cdot \hat{d} = (x_b \cdot d_x) \oplus (r_b \cdot d_r)$$

which implies that

$$\hat{d} \cdot (\hat{x}_0 \oplus \hat{x}_1) = (d_x \cdot (x_0 \oplus x_1)) \oplus (d_r \cdot (r_0 \oplus r_1)).$$

Note that the output of $\hat{\mathcal{A}}$ is a tuple $(b, \hat{x}_b, \hat{d}, \hat{d} \cdot (\hat{x}_0 \oplus \hat{x}_1))$. One can now define a new QPT adversary $\mathcal{A}$ which runs $\hat{\mathcal{A}}$ and then outputs $(b, x_b, d_x, \hat{d} \cdot (\hat{x}_0 \oplus \hat{x}_1) \oplus (d_r \cdot (r_0 \oplus r_1)))$. This then implies that

$$\Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k) \in H_k] - \Pr_{(k,t_k) \leftarrow \mathrm{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k) \in \overline{H}_k] \;\; \geq p(\lambda) \; .$$

Hence, the adaptive hardcore bit of the original NTCF family is violated. We conclude that the randomized encoding must also satisfy the adaptive hardcore bit property.

□

3.3 Resource estimation

In this section, we give some estimates of the resources required to run our modified protocols. We summarize this information in Table 1 and proceed to explain the results. The functions listed in the table are the same as the ones from [KMCVY22], as these are the existing candidate TCFs used in proof of quantumness protocols.

Function	Adaptive H.C.	# of quantum-classical interleavings	Depth	Width
LWE	✓	3	14	$O(\lambda l^4)$
Ring-LWE	✗	4	18	$O(\lambda l^4)$
$x^2 \mod n$	✗	4	18	$O(\lambda l^4)$
Diffie-Hellman	✗	4	18	$O(\lambda l^4)$

Table 1: The table of resource estimations for each type of (N)TCF function that may be used. Here H.C. means hardcore bit. The number of quantum-classical interleavings refers to the instances where the prover performs a constant-depth quantum circuit followed by a classical computation. This is done, for instance, in the preparation of cat states as well as when it responds to one of the verifier's challenges. Depth refers to the total number of layers of quantum gates that the prover has to perform. Width refers to the width of the quantum circuits the prover has to implement. Here, λ denotes the security parameter and l is the size of the branching program implementing the randomized encoding, as described in Appendix A.

3.3.1 Quantum depth and quantum-classical interleavings

In this subsection we explain the overall quantum depth that the prover has to perform in our modified proofs of quantumness. Depth here represents the number of layers of quantum gates or measurements (as described in Section 2) that the prover will perform throughout the protocol, in the worst case. As mentioned, the prover's operations consist of alternating between constant-depth quantum circuits and log-depth classical computation. This latter step we referred to as a quantum-classical interleaving.

For the NTCF-based protocol which uses LWE, the total quantum depth is 14 and 3 quantum-classical interleavings are performed, whereas for the TCF-based approaches the depth is 17 and the number of interleavings is 4. Let us explain where these numbers come from:

1. **Preparation of cat states.** As mentioned, we prepare cat states by interleaving a constant depth quantum circuit with a log-depth classical computation, followed by another quantum

circuit. The exact steps are outlined in [WKST19], while here we just summarize the gates performed in each step. The procedure starts with a layer of Hadamard gates followed by two layers of $CNOT$ gates. Some of the qubits are then measured in the computational basis. The remaining qubits will collapse to a poor man's cat state, while the measured qubits contain the parity information for that state. To "correct" the state to a cat state, the parity information is used to compute a Pauli-X correction. This is one quantum-classical interleaving. The final quantum layer consists of Pauli-X gates. Thus, the total depth will be 5 and we have 1 quantum-classical interleaving. This applies to all cat states, as they can be prepared in parallel.

2. **Evaluation of the randomized encoded function.** As illustrated can see in Figure 8, the classical circuit for a randomized encoding has depth 3. In the quantum case, the AND gates are implemented by Toffoli gates and the XOR gate is a $CNOT$. As the quantum gates are reversible, one needs to *uncompute* any auxiliary results and so the quantum depth will be double that of the classical circuit. Hence, for this step the quantum depth is 6 and there are no quantum-classical interleavings.

3. **Measurement of the Y register.** Measuring the image register requires a layer of computational basis measurements and so the depth is 1. The results are read out and sent to the verifier, which we count as 1 quantum-classical interleaving.

4. **Preimage test or equation/Bell test.** If a preimage test is performed, the prover only needs to measure the X register in the computational basis and report the result. This counts as depth 1 and 1 interleaving. In the NTCF protocol, if an equation test is performed, then the prover is expected to apply a layer of Hadamard gates to the X register and measure them. This counts as depth 2 and 1 interleaving. In the TCF protocol, when the computational Bell test is performed, the prover's operations (as outlined in Subsection 3.1) will consist of a layer of CZ gates, a layer of Hadamard gates together with a computational basis measurement, a classical computation and reporting the results to the verifier, a Pauli-Z operation, a rotation gate and finally another measurement and reporting the results to the verifier. This counts as depth 6 and 2 interleavings.

We can see that for the NTCF-based protocol the worst-case depth is $5+6+1+2 = 14$ and the number of interleavings is $1+0+1+1 = 3$. For the TCF-based one, the depth is $5+6+1+6 = 18$ and the number of interleavings is $1+0+1+2 = 4$.

3.3.2 Circuit width

The constant-depth versions of the proof of quantumness protocols require larger numbers of qubits than the original version. As explained, most of this is due to the use of cat states, which effectively copy the input and allow us to apply a constant depth circuit with bounded fan-out gates. That circuit is a randomized encoding of the original TCF. Following the construction of randomized encodings from [AIK04] and described in Appendix A, the width of the constant-depth circuit will depend on the size of the branching program used to evaluate the original function. In Appendix A we explain how, as a result of *Barrington's theorem*, the size of this branching program is exponential in the depth of the original TCF. As all TCFs considered here can be evaluated in logarithmic depth, the resulting branching programs will have sizes polynomial in the security parameter λ. Giving a precise account of the size of the branching program, as a function of λ, for each TCF, is beyond the scope of this paper. Instead, we find in Appendix A that the overall circuit width for the prover's quantum circuit is $O(\lambda l^4)$, where l is the size of the branching program used to evaluate the TCF. The λ factor comes from having to repeat the branching program construction in parallel $O(\lambda)$ times. This is because one branching program computes a single output bit of the TCF and so one has to consider a different branching program (of the same size) for each output bit.

As a rough estimate, we can relate the width to the security parameter for the LWE-based NTCF of [Mah18, BCM$^+$18]. There we know from [GH20] that the functions can be evaluated in depth $\propto 4 \log \lambda$. From Barrington's theorem, the size l of the corresponding branching program is on the order of λ^8. As the width is $O(\lambda l^4)$, we find that the prover requires $O(\lambda^{33})$ qubits. This is a discouraging result for the purposes of implementing these protocols on near-term devices. However, it should be noted that this was merely a rough calculation based on existing asymptotic estimates. We conjecture that these estimates are not optimal and can be improved with a tighter analysis, better circuit implementations and more compact branching programs. Additionally, for a fixed-size implementation (say $\lambda = 50$), it is likely that additional optimizations are possible that could further reduce the number of required qubits.

4 Proofs of quantumness via phase encoding

The first construction based on randomized encoding is a generic method that works for all types of (N)TCFs. However, as mentioned, its naive implementation based on Barrington's theorem leads to circuits which are too wide to be implemented on near-term devices.

In this section, we propose another approach that can be implemented on much narrower circuits, thus bringing it closer to implementation on near-term devices. This construction relies on *phase encodings* to evaluate a specific NTCF, based on the LWR problem that is defined in Subsection 2.2. As we will see, the resulting circuits also involve only constant quantum depth and logarithmic classical depth.

Before presenting the protocol, we first define the LWR-based NTCF, denoted as f, and introduce its phase encoded implementation.

4.1 LWR-based NTCF

The LWR-based NTCF was suggested in [BCM+18] but not used. It is however used in [ZKML+21], but without the phase encoding. The specific NTCF we consider is the following:

Definition 4.1 (LWR-based NTCF)**.** Let $\lambda > 0$ be a security parameter. We take $n(\lambda), m(\lambda), q(\lambda), p(\lambda)$ as functions of λ subject to the following constraints: $n = O(\lambda)$, $q = 2^{O(n)}$ is prime, $m = \Omega(n \log q)$, and $p = O(\sqrt{mn \log q})$ is a power of 2. Additionally χ will denote a discrete Gaussian distribution over $\mathbb{Z}_q$ having width $O(q/p^5)$. Taking $\mathbf{A} \leftarrow_r \mathbb{Z}_q^{m\times n}$, $s \leftarrow_r \{0,1\}^n$, $e \leftarrow_{\chi^m} \mathbb{Z}_q^m$ (so that $\|e\|_\infty = O(q/p^5)$), we define the function

$$f(b,x) : \{0,1\} \times \mathbb{Z}_q^n \to \mathbb{Z}_p^m = \lfloor g(b,x) \rceil_p$$

where

$$g(b,x) : \{0,1\} \times \mathbb{Z}_q^n \to \mathbb{Z}_q^m = \mathbf{A}x + b \cdot (\mathbf{A}s + e).$$

For the specific constants in the parameters defined above, we use the same values as in [BCM+18]. It should be noted that the width of the error distribution is taken to be polynomially smaller than in [BCM+18] ($O(q/p^5)$ versus $O(q/p)$). But since the width is still superpolynomial (in n) we are still in the "hardness regime" where both LWE and LWR are intractable. For more details, we refer the reader to the Preliminaries of [BCM+18]. The reason for this choice will become apparent in Subsection 4.3.1.

Although we are referring to f as an NTCF, it is not clear if this is indeed the case. Following the definition from Subsection 2.3, we next show that all the properties are satisfied. As $f(b,x)$ uses the same LWE instance as the LWE-based NTCF of [BCM+18], we will have the same Gen, which immediately proves the efficient function generation property. Additionally, Lemma 2.1 confirms that the (k, t_k) pair sampled by Gen is also the key and trapdoor pair for the LWR-based function (for this reason we will sometimes write the function as f_k). We can also see that if $(0,x)$ is the preimage of $y = f(0,x)$, the other preimage is $(1, x - s)$. The trapdoor injective pair property then follows. The efficient evaluation property comes from the fact that mod-q matrix multiplication and additions can be efficiently performed by polynomial-depth quantum circuits. In fact, the rest of this section is devoted to showing an efficient evaluation in constant quantum-depth using the phase encoding construction.

We are left with showing the adaptive hardcore bit property. As a first step, we show the following:

Lemma 4.1. x_0 and x_1 form a claw of the LWR-based NTCF if and only if they are also a claw of the corresponding LWE-based NTCF (from [BCM+18]), with high probability.

Proof. Consider

$$\begin{aligned} f(b,x) &= \lfloor \mathbf{A}x + b\cdot(\mathbf{A}s+e) \rfloor_p \\ h(b,x) &= \mathbf{A}x + b\cdot(\mathbf{A}s+e) + e' \end{aligned}$$

where h is the LWE-based NTCF using in [BCM+18] and both functions are based on *the same LWE sample* $\mathbf{A}s+e$. The statement we would like to show is then re-expressed as

$$f(0,x_0) = f(1,x_1) \Leftrightarrow h(0,x_0) = h(1,x_1)$$

with high probability over the choices of $\mathbf{A}$, s, and e. We can prove it by showing both implications.

- ($\rightarrow$) Consider its contrapositive: if $h(0,x_0) \neq h(1,x_1)$, then $f(0,x_0) \neq f(1,x_1)$, with high probability. In [BCM+18], it was shown that $h(0,x_0) \neq h(1,x_1)$ if and only if $x_1 \neq x_0 - s$, with high probability. Now take $x_1 = x_0 - s + w$ for some non-zero $w \in \mathbb{Z}_q^n$. We know that $\mathbf{A}w$ is a uniformly random vector (over the random choice of $\mathbf{A}$) and therefore every bit of $f(1,x_1)$ has a probability of $\frac{1}{2}$ to be flipped with respect to $f(0,x_0)$. Thus, the probability of $f(0,x_0) = f(1,x_1)$ can be bounded by the additive Chernoff inequality

  $$\Pr(d_H(f(0,x_0), f(1,x_1)) = 0) \leq \exp \quad -\frac{m \log_2 p}{4}$$

  which is negligible.

- ($\leftarrow$) Suppose $h(0,x_0) = h(1,x_1)$, which immediately leads to $x_1 = x_0 - s$, with high probability. We then have $f(0,x_0) = \lfloor \mathbf{A}x_0 \rfloor_p$ and $f(1,x_1) = \lfloor \mathbf{A}x_0 + e \rfloor$. As we have $\|e\|_\infty = O(q/p^5)$, the probability of $f(0,x_0) = f(1,x_1)$ is $1 - \mathit{negl}(n)$ as shown in [AKPW13].

□

Now we have all the ingredients for the proof of the adaptive hardcore bit property.

Theorem 4.1. The LWR-based NTCFs $(f_k(b,x))$ have the adaptive hardcore bit property.

Proof. We present a proof by contradiction. Suppose $f_k(b,x) = \lfloor \mathbf{A}x + b(\mathbf{A}s+e) \rfloor_p$ is an LWR-based NTCF where k is the key and t_k is the trapdoor, both generated by Gen. Assume there exists a QPT adversary $\hat{\mathcal{A}}$ that breaks the adaptive hardcore bit property of f. This means that there exists a non-negligible function $\kappa(m)$ that satisfies

$$\Pr_{(k,t_k)\leftarrow \textsc{Gen}_{\mathcal{F}}(1^\lambda)}[\hat{\mathcal{A}}(k) \in \hat{H}_k] - \Pr_{(k,t_k)\leftarrow \textsc{Gen}_{\mathcal{F}}(1^\lambda)}[\hat{\mathcal{A}}(k) \in \overline{\hat{H}}_k] \;\; \geq \kappa(m)$$

where

$$\begin{aligned} \hat{H}_k &= \quad (b, x_b, d, d\cdot(x_0 \oplus x_1)) \,|\, b \in \{0,1\},\ (x_0,x_1) \in \hat{\mathcal{R}}_k \quad , \\ \overline{\hat{H}}_k &= \{(b, x_b, d, c) \,|\, (b, x, d, c \oplus 1) \in \hat{H}_k \quad , \end{aligned}$$

and $\hat{\mathcal{R}}_k$ is the set of all tuples x_0, x_1 such that $f_k(0,x_0) = f_k(1,x_1)$. We can then consider the LWE-based NTCF $h_k(b,x) := \mathbf{A}x + b\cdot(\mathbf{A}s+e) + e'$, whose corresponding sets are denoted by H_k, $\overline{H}_k$, and $\mathcal{R}_k$. As is shown in Lemma 4.1, we have $\mathcal{R}_k = \hat{\mathcal{R}}_k$, with overwhelming probability, hence $H_k = \hat{H}_k$ and $\overline{H}_k = \overline{\hat{H}}_k$. Therefore, we can define the QPT adversary, $\mathcal{A} := \hat{\mathcal{A}}$. It satisfies

$$\Pr_{(k,t_k)\leftarrow \textsc{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k) \in H_k] - \Pr_{(k,t_k)\leftarrow \textsc{GEN}_{\mathcal{F}}(1^\lambda)}[\mathcal{A}(k) \in \overline{H}_k] \;\; \geq \kappa(m)$$

which breaks the adaptive hardcore bit property of LWE-based NTCFs. □

This implies that $f(b,x)$ satisfies all requirements of an NTCF.

4.1.1 Prime q

As mentioned in Definition 4.1, we require q to be a prime. This is, in fact, also a requirement in [BCM$^+$18]. The reason for this is that some of the properties of these NTCF-based constructions hold only when $\mathbb{Z}_q$ is a finite field, rather than a finite ring. Normally, this would just be a minor technical point. However, in our case since we would like to perform the prover's operations in constant depth, we would need to provide a procedure that allows the prover to prepare equal superpositions over the field elements. In other words, the prover needs to create an equal superposition of a prime number of elements. While this can be done in constant quantum depth, using cat states and ideas from [HŠ05], we will find that this is not necessary, provided q is sufficiently large and *sufficiently close to a power of 2*. In this section, we show that these conditions can indeed be satisfied and it is possible to efficiently choose a prime q that is close to a power of 2.

We start with a result from [Dus98]:

Lemma 4.2 ([Dus98])**.** For $q' > 3275$, there exists a prime q in the interval

$$q' < q < \quad 1 + \frac{1}{2\ln^2 q'} \quad q'.$$

This implies that the ratio of q and $q' = 2^n$ is bounded by

$$1 < \frac{q}{q'} < 1 + \frac{1}{2(\ln 2)^2 n^2} = 1 + O(n^{-2}).$$

Moreover, a specific prime in between $q' = 2^n$ and $\quad 1 + \frac{1}{2\ln^2 q'} \quad q'$ can be efficiently found. It suffices to sample random integers in the range and check if they are prime. The checking can be done by (for instance) the Miller-Rabin algorithm [Rab80], in polynomial time. We can show that the number of samples to check is $O(n)$ using the *Prime number theorem*, which states that, if $\pi(N)$ is the prime counting function, for integers in the range $(0, N)$, then it is the case that

$$\pi(N) \sim \frac{N}{\log N}.$$

Thus, the number of primes in the desired range can be estimated by

$$\pi \quad 1 + \frac{1}{2\ln^2 q'} \quad q' \quad \sim \frac{2^n \quad 1 + \frac{1}{2(\ln 2)^2 n^2}}{n + \log \quad 1 + \frac{1}{2(\ln 2)^2 n^2}} \sim 2^n \quad \frac{1}{n} + \frac{1}{2(\ln 2)^2 n^3} + O(n^{-4})$$

and

$$\pi \quad 1 + \frac{1}{2\ln^2 q'} \quad q' \quad - \pi(q') = \frac{2^n}{2(\ln 2)^2 n^3} + O(2^n n^{-4}).$$

Therefore, the density of primes in the range is

$$\rho = \frac{\pi \quad 1 + \frac{1}{2\ln^2 q'} \quad q' \quad - \pi(q')}{q' \frac{1}{2\ln^2 q'}} \sim \frac{2^n \frac{1}{2(\ln 2)^2 n^3}}{2^n \frac{1}{2(\ln 2)^2 n^2}} = \frac{1}{n} + O(n^{-2}),$$

which immediately implies that a prime can be found with an expected number of $O(n)$ random samples. All of this is incorporated in the Gen procedure as that is responsible for choosing a suitable q. As will also be mentioned later, since q is close to a power of 2, when the prover has to create an equal superposition over the elements of $\mathbb{Z}_q$ it will instead create the superposition over elements up to q', the nearest power of 2, larger than q. The resulting state will be sufficiently close in trace distance that we only incur a $1/poly(n)$ penalty in completeness for making this replacement.

4.2 Phase encoding

The concept of phase encoding was described in Section 2. In this section we will look at several properties of the phase encoding for the LWR-based NTCF (Definition 4.1). We aim to show how to evaluate $g(b, x) = \mathbf{A}x + b \cdot (\mathbf{A}s + e)$ in phase, and show that measuring the resulted state in Hadamard basis will reveal the value of $f(b, x) = \lfloor g(b, x) \rceil_p$, with high probability.

It is natural to start by considering the phase encoding of $g(b, x)$ for a specific (b, x). Note that $x \in \mathbb{Z}_q^n$ and $g(b, x) \in \mathbb{Z}_q^m$, both being vectors. The phase encoded state that we would like the prover to prepare (for each b and x) should have the following form:

$$|\phi(b, x)\rangle = \bigotimes_{i=1}^{m} |\phi_i(b, x)\rangle \tag{22}$$

with

$$|\phi_i(b, x)\rangle = \frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{i\phi_i(b,x)} |\bar{1}\rangle) \tag{23}$$

and

$$\phi_i(b, x) = \frac{2\pi g_i(b, x)}{q} - \frac{\pi}{2} \tag{24}$$

where g_i represents the i'th component of $g(b, x)$.

For the majority of this section, we will focus on the case $p = 2$. That is, we assume that $f(b, x)$ simply takes the most significant bit of each component of $g(b, x)$. This, of course, is not the NTCF we defined since there we had that $p = O(\sqrt{mn \log q})$. We will address the case of general p in Subsection 4.2.3.

For $p = 2$, we denote the output of $f(b, x) = \lfloor g(b, x) \rceil_2$ by y, a binary string of length m. We have $y_i = \lfloor g_i(b, x) \rceil_2$ where y_i is the i'th bit of y and $g_i(b, x)$ is the i'th component of $g(b, x)$. Before explaining how to prepare the phase encoded state in constant depth, let us first investigate how to decode $y = f(b, x)$ from $|\phi(b, x)\rangle$ with high probability.

4.2.1 Decoding by measurements

The phase encoding can be *probabilistically decoded* through Hadamard measurements. Denote the process of measuring the $XX...X$ observable on the state in Equation 25 by M and the measurement outcomes of all m phase encoded states by $z \in \{0, 1\}^m$. One can then write $z \leftarrow M(|\phi(b, x)\rangle)$. It should be clear that $z = y$ indicates that the decoding was completely successful.

Let us consider the case of a single component in the encoding, namely $|\phi_i\rangle$. In order to investigate the possible values of $z_i = M(|\phi_i\rangle)$, $|\phi_i\rangle$ can be rewritten as

$$\begin{aligned} |\phi_i\rangle &= \frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{i\phi_i} |\bar{1}\rangle) & (25) \\ &= \frac{1}{2}((1 + e^{i\phi_i}) |\bar{+}\rangle + (1 - e^{i\phi_i}) |\bar{-}\rangle). & (26) \end{aligned}$$

If the qubit is measured in the Hadamard basis, we can express the outcome probabilities as

$$\mathrm{Pr}_{\mathrm{M}}(\pm | \, |\phi_i\rangle) = \frac{1}{4}[(1 \pm \cos \phi_i)^2 + \sin^2 \phi_i] = \frac{1}{2}(1 \pm \cos \phi_i). \tag{27}$$

with $\phi_i = \frac{2\pi g_i}{q} - \frac{\pi}{2}$. Note that $g_i < q/2$ is equivalent to $y_i = \lfloor g_i \rceil_2 = 0$. Additionally, $g_i < q/2$ leads to $\cos \phi_i > 0$. Therefore the probability of getting $+$ is larger than that of $-$. If we map $+$ to 0 and $-$ to 1, it is clear that the Hadamard measurement is essentially a probabilistic decoding of y_i from ϕ_i, with success probability always greater than $\frac{1}{2}$. More compactly, we can write the probability of measuring any z_i from $|\phi_i\rangle$ by

$$\mathrm{Pr}_{\mathrm{M}}(z_i | \, |\phi_i\rangle) = \frac{1}{2}(1 + (-1)^{z_i} \cos \phi_i). \tag{28}$$

Furthermore, the probability of *successfully decoding* ϕ_i (i.e. $z_i = y_i$) is denoted by

$$p_{\mathrm{cor}}(\phi_i) := \Pr(z_i = y_i) = \mathrm{Pr}_{\mathrm{M}}(y_i | \, |\phi_i\rangle). \tag{29}$$

where $\Pr(z_i = y_i) = \Pr_{\mathrm{M}}(+|\phi_i) = \frac{1}{2}(1+\cos\phi_i)$ if $y_i = 0$ and $\Pr(z_i = y_i) = \Pr_{\mathrm{M}}(-|\phi_i) = \frac{1}{2}(1-\cos\phi_i)$ if $y_i = 1$. Similarly, the probability of unsuccessful decoding is represented by

$$p_{\mathrm{inc}}(\phi_i) := \Pr(z_i \neq y_i) = \Pr_{\mathrm{M}}(\neg y_i|\,|\phi_i\rangle) = 1 - p_{\mathrm{cor}}(\phi_i). \tag{30}$$

We can now evaluate the expected values of these probabilities over the uniform choice of the matrix $\mathbf{A}$ and show the following:

Lemma 4.3. Over the choice of matrix $\mathbf{A}$, the average probability of successful decoding of any $|\phi_i\rangle$ is $\frac{1}{2} + \frac{1}{\pi} \approx 0.82$.

Proof. To clarify, there are two sources of randomness here. On the one hand we have the randomness of the measurement and on the other hand we have the random choice of the matrix $\mathbf{A}$. We're interested in seeing the expected probability of a successful (as well as an unsuccessful) decoding over the choice of $\mathbf{A}$. As $g(b,x) = \mathbf{A}x + b\cdot(\mathbf{A}s+e)$, we can see that if $\mathbf{A}$ is uniform (over a finite field), then $g(b,x)$ will also be uniform (for any non-zero b and x). Hence, $\Pr(\phi_i) = \Pr(g_i) = \frac{1}{q}$ for all $\phi_i \in \{-\frac{\pi}{2}, \frac{2\pi}{q} - \frac{\pi}{2}, \ldots, \frac{3\pi}{2}\}$. The expected probability of a correct decoding is then

$$\begin{aligned}\bar{p}_{\mathrm{cor}} := \mathbb{E}_{\mathbf{A}}(p_{\mathrm{cor}}(\phi_i)) &= \sum_{g_i=0}^{q/2-1} \Pr(\phi_i)\Pr_{\mathrm{M}}(+|\phi_i) + \sum_{y_i=q/2}^{q-1} \Pr(\phi_i)\Pr_{\mathrm{M}}(-|\phi_i) \\ &= 2\sum_{g_i=0}^{q/2-1} \Pr(\phi_i)\Pr_{\mathrm{M}}(+|\phi_i) \\ &= 2\sum_{g_i=0}^{q/2-1} \frac{1}{q}\frac{1}{2}(1+\cos\phi_i) := S\end{aligned} \tag{31}$$

which we can view as a *Riemann sum.* For large q, the summation converges to an integral

$$\bar{p}_{\mathrm{cor}} = S \to I := 2\int_0^{\frac{q}{2}-1} \frac{1}{2q}\left(1+\cos\left(\frac{2\pi g_i}{q} - \frac{\pi}{2}\right)\right) dg_i. \tag{32}$$

By the change of variable $\phi_i = \frac{2\pi g_i}{q} - \frac{\pi}{2}$, this becomes

$$\bar{p}_{\mathrm{cor}} \to I = \frac{1}{\pi}\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{1}{2}(1+\cos(\phi_i))d\phi_i \tag{33}$$

$$= \frac{1}{2} + \frac{1}{\pi} \sim 0.82. \tag{34}$$

We also have the expected probability of an incorrect decoding

$$\bar{p}_{\mathrm{inc}} := \mathbb{E}_{\mathbf{A}}(p_{\mathrm{inc}}(\phi_i)) \to 1 - \bar{p}_{\mathrm{cor}} = \frac{1}{2} - \frac{1}{\pi} \sim 0.18. \tag{35}$$

The approximation $S \to I$ comes with an error which we can bound. Such an error for an $(l+1)$-order differentiable integrand χ can be determined with the Euler-Maclaurin formula

$$S - I = \sum_{k=1}^{l} \frac{B_k}{k!}\left(\chi^{(k-1)}\left(\frac{q}{2}-1\right) - \chi^{(k-1)}(0)\right) + R_l \tag{36}$$

where B_k is the k-th Bernoulli number, $R_l = o(q^{-l})$ is the remainder term, and $\chi(y_i) = \frac{1}{q}(1 + \cos(\frac{2\pi g_i}{q} - \frac{\pi}{2}))$ is the integrand. We can see that $\chi^{(k-1)}(\frac{q}{2}-1) - \chi^{(k-1)}(0) = 0$ for odd k. Therefore, the error can be written as

$$\begin{aligned} S - I &= \frac{B_2}{2}\frac{1}{q}\frac{2\pi}{q}\left(-\sin\left(\frac{\pi}{2} - \frac{2\pi}{q}\right) + \sin\left(-\frac{\pi}{2}\right)\right) + o(q^{-2}) \\ &= -\frac{1}{3q^2} + o\left(q^{-2}\right) = O(q^{-2}).\end{aligned} \tag{37}$$

□

As $g(b,x)$ is uniform (over the random choice of $\mathbf{A}$ and whenever $(x,b) \neq (0,0)$), each of its components will be a uniform value in $\mathbb{Z}_q$. Thus, we can view the measurement of each component of $|\phi(b,x)\rangle$ to be an independent and identically distributed random variable. As the expected probability of a correct decoding is 0.82, it follows from a Chernoff bound that $0.82m$ values will be decoded correctly, with overwhelming probability over the choice of $\mathbf{A}$. While this means that most values are correctly decoded, we, in fact, need *all* values to be decoded correctly with high probability. To achieve this, we use a *classical repetition code* and repeat each output component several times in order to take a majority vote.

4.2.2 Decodability and repetition code ($p = 2$)

Instead of the prover having to prepare $|\phi(b,x)\rangle$ (for each b and x), we will instead ask it to prepare:

$$|\phi(b,x)\rangle = \bigotimes_{i=1}^{n} |\phi_i(b,x)\rangle^{\otimes v} = \bigotimes_{i=1}^{n} \left(\frac{1}{\sqrt{2}}(|0\rangle + e^{i\phi_i}|1\rangle) \right)^{\otimes v} \tag{38}$$

where v represents the number of repetitions. In this case, to decode the value of the i'th component, one measures all v copies of that component and uses the majority outcome as the value z_i.

We say that one component, for instance the i'th component, has been correctly decoded, if $z_i = y_i$, where recall that y_i is the most-significant bit of $g_i(b,x)$. By analogy, we say that the whole state has been correctly decoded if all of its components were (i.e. $z = y$). Our goal is to find the relation between v and m such that $z = y$ with sufficiently high probability (say, 99%) *for most states* $|\phi(b,x)\rangle$ (say, 99% of all such states). In doing so, we show the following

Theorem 4.2. At least 99% of all $|\phi(b,x)\rangle$ states can be correctly decoded with probability 99%, whenever $v = \Omega(m^2 \log m)$.

Proof. Without loss of generality, we focus on the case of $g_i < \frac{q}{2}$, that is $y_i = 0$. Recall that

$$p_{\mathrm{cor}}(\phi_i) = \Pr_{\mathrm{M}}(+|\phi_i) = \frac{1}{2}(1+\cos(\phi_i)) = \frac{1}{2}\left(1 + \sin\left(\frac{2\pi g_i}{q}\right)\right). \tag{39}$$

It should be clear that for the very special case $g_i = 0$, the probability of having the correct measurement outcome is $\frac{1}{2}$. In this case, it is impossible to tell if z_i should be 0 or 1 even with repetition, because no matter how large v is, there will always be an equal number of correctly and incorrectly decoded bits, on average. Therefore, any component g_i that is extremely close to 0 or $\frac{q}{2}$ so that $p_{\mathrm{cor}}(\phi_i)$ is close to $\frac{1}{2}$ would make the whole $|\phi(b,x)\rangle$ state *undecodable*[11].

To be more explicit, we will consider $|\phi_i\rangle$ to be undecodable whenever we have that either $|g_i| < \frac{q}{cm}$ or $|g_i - q/2| < \frac{q}{cm}$, for a constant $c > 0$ to be determined later. But as noted before, for a uniform $\mathbf{A}$, each g_i (excluding the case $g(0,0)$) is also uniform in $\mathbb{Z}_q$. It follows that the probability that g_i leads to an undecodable $|\phi_i\rangle$ is at most $\frac{1}{q}\frac{4q}{cm} = \frac{4}{cm}$, over the choice of $\mathbf{A}$. From a union bound, we then also have that the probability of $|\phi(b,x)\rangle$ to be undecodable (i.e. at least one of its components is undecodable) is at most $m\frac{4}{cm} = \frac{4}{c}$. This means that at least a fraction $1 - \frac{4}{c}$ of all $|\phi(b,x)\rangle$ states are, in fact, decodable. That is, all of their components are at least $\frac{q}{cm}$ away from the undecodability boundary. By taking $c = 400$, we have that 99% of $|\phi(b,x)\rangle$ are decodable.

Without loss of generality, let's now consider a state that is barely decodable, with say $g_i = \frac{q}{cm}$. The probability of correctly decoding the corresponding $|\phi_i\rangle$ state will be

$$p_{\mathrm{cor}}(\phi_i) = \frac{1}{2}\left(1 + \sin\left(\frac{2\pi g_i}{q}\right)\right) \approx \frac{1}{2}\left(1 + \frac{1}{O(m)}\right). \tag{40}$$

[11] In fact, even if we ignore the cases where $p_{\mathrm{cor}}(\phi_i) = \frac{1}{2}$, it is still required to have $v = O(q)$ to distinguish between $\phi_i = \frac{2\pi}{q} - \frac{\pi}{2}$ and $\phi_i = -\frac{2\pi}{q} + \frac{3\pi}{2}$ where $g_i = 1$ and $g_i = q - 1$, respectively. This is clearly unacceptable since q is exponential in n and the resulting circuit would be exponentially wide.

The state is biased away from $1/2$ by $1/O(m)$. From an application of the Chernoff-Hoeffding bound[12] it follows that repeating the measurement $\Omega(m^2)$ times and taking a majority vote is enough to ensure that the value is correctly decoded with *constant probability* (say 99%). Of course, we want that *all* m values are correctly decoded which means that we should take the number of repetitions v so that the probability of correctly decoding one value is at least $1 - 1/O(m)$. Once again, we can use Chernoff-Hoeffding and find that $v = \Omega(m^2 \log m)$. As the probability of incorrectly decoding one value is now $1/O(m)$, from a union bound the probability of incorrectly decoding *any* of the m values is $O(1)$. By suitably choosing the constant factors, we can set this probability to be, say 1%. We therefore have that $v = \Omega(m^2 \log m) = \Omega(n^2 \log m \log^2 q) = \Omega(n^4 \log n)$. □

4.2.3 Phase encoding for general p

The analysis from the previous subsections was concerned with the case $p = 2$. We now adapt this to the general case of $p = O(\sqrt{mn \log q})$.

As we expect p to be a power of 2, the rounding $\lfloor g_i \rceil_p$ for any value of g_i is exactly a $(\log_2 p)$-bit number. What we have been doing so far with the phase encoding is to encode the most significant bit of $f_i = \lfloor g_i \rceil_p$ in phase. What about the other $\log_2 p - 1$ bits? The solution is simply to phase encode those bits as well.

Lemma 4.4. Applying the phase encoding to the $\log_2 p$ significant bits of every $g_i \in \mathbb{Z}_q$, leads to a repetition factor $v = \Omega(n^4 \log^2 n)$ in order to achieve the same guarantees as Theorem 4.2.

Proof. Specifically, the k'th significant bit of g_i can be encoded as

$$|\phi_{i,k}\rangle = \frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{i\phi_{i,k}}|\bar{1}\rangle) \tag{41}$$

with

$$\phi_{i,k} = \frac{2^k \pi g_i}{q}. \tag{42}$$

How does this affect the decodability results of the previous sections? The expected probability of decoding a single bit, without repetition, will still be negligibly close to 0.82. This is because, as we saw in Subsection 4.2.1, the deviation from this expectation is inverse in the square of the field size, which is now $\sim \frac{q}{2^k}$. As $k \leq \log_2 p$, $p = O(\sqrt{mn \log q})$ so that $2^k = O(\sqrt{mn \log q})$ and $q = 2^{O(n)}$, the deviation from the expected value of 0.82 remains negligible in n (or λ).

The decodability boundary, from Subsection 4.2.2, also changes from $\frac{q}{cm}$ to $\frac{q}{2^k cm}$. As $2^k = O(\sqrt{mn \log q})$ and $m = \Omega(n \log q)$, the boundary becomes $\frac{q}{c' n^4}$, for some constant $c' > 0$. Following the same steps as in Subsection 4.2.2, to ensure that most states can be correctly decoded, we see that the number of repetitions remains $\Omega(n^4)$. But this is just for the m-bit vector containing the k'th most significant bit of each component. As we have $\log_2 p$ such vectors, and we want all of them to be decoded correctly, we need to add an additional $\log_2 p$ factor so that overall we have $v = \Omega(n^4 \log n \log_2 p) = \Omega(n^4 \log^2 n)$. □

Thus, for each b and x, the state the prover will prepare is

$$|\phi(b, x)\rangle = \bigotimes_{i=1}^{n} \bigotimes_{k=1}^{\log_2 p} \left(|\bar{0}\rangle + e^{i\phi_{i,k}}|\bar{1}\rangle \right)^{\otimes v}. \tag{43}$$

4.2.4 Constant-depth circuit implementation

Here we show that the phase encoding construction can be performed in constant quantum depth.

[12] Each measurement is viewed as an i.i.d. random variable. The empirical mean of these variables is expected to be close to $1/2 + 1/O(m)$. Chernoff-Hoeffding tells us that a deviation of ϵ from this expected value occurs with probability $\exp(-v\epsilon^2)$. Thus, since the case of interest is $\epsilon = 1/O(m)$, we can see that to have a constant probability of incorrectly decoding, it must be that $v = \Omega(m^2)$.

Theorem 4.3. It is possible to prepare the state in Equation 43 in constant quantum depth and with logarithmic depth classical computation.

Proof. We've already mentioned that cat states can be prepared in constant quantum depth with one quantum-classical interleaving. Let us then assume that we have sufficient cat states (of a size that will be determined later) and see how we can apply the required phases in constant quantum depth.

Recall that $g(b, x) = \mathbf{A}x + b \cdot (\mathbf{A}s + e)$, and determines the phase[13] $\phi_i = \frac{2\pi g_i}{q} - \frac{\pi}{2}$. The phase can then be expressed as

$$\begin{aligned} e^{i\phi_i} &= \exp\left(-\frac{\pi i}{2}\right) \exp\left(bi\frac{2\pi(\mathbf{A}s)_i + 2\pi e_i}{q}\right) \exp\left(\frac{2\pi i}{q}\sum_{j=1}^{n} A_{ij}x_j\right) \\ &= \exp(\phi_i'(b)) \prod_{j=1}^{n} \exp\left(\frac{2\pi i}{q} A_{ij}x_j\right) \end{aligned} \tag{44}$$

where

$$\exp(\phi_i'(b)) := \exp\left(-\frac{\pi i}{2}\right) \exp\left(bi\frac{2\pi(\mathbf{A}s)_i + 2\pi e_i}{q}\right). \tag{45}$$

Note that ϕ_i' only depends on b and not on x. Having multiple copies of b, we can easily apply a ϕ_i' rotation in parallel using Z-rotations (R_z) and controlled-Z-rotations (CR_z):

$$R_z\left(-\frac{\pi}{2}\right) CR_z\left(\frac{2\pi(\mathbf{A}s)_i + 2\pi e_i}{q}\right)\left(\frac{1}{\sqrt{2}}(|\bar{b}\rangle|\bar{0}\rangle + |\bar{b}\rangle|\bar{1}\rangle)\right) = |\bar{b}\rangle \otimes \frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{i\phi_i'(b)}|\bar{1}\rangle). \tag{46}$$

The corresponding circuit is shown in Figure 4.

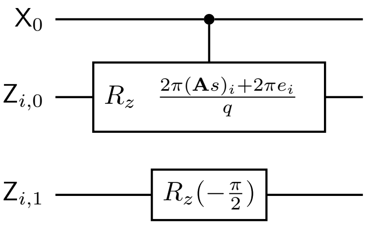

Figure 4: The quantum circuit for the vector addition operations in phase encoding. Here X_0 is the first qubit of the X register that stores information of b. $\mathsf{Z}_{i,j}$ is the j'th qubit of the i'th cat state which stores information of ϕ_i.

We now need to implement the phase-encoded matrix-vector multiplication in parallel on the cat state. Note that x_j is a non-negative integer less than q and it can be expanded as

$$x_j = \sum_{k=0}^{\lceil \log_2(q) - 1 \rceil} 2^k x_{j,k} \tag{47}$$

denoting the k'th significant bit of x_j by $x_{j,k}$. The phase can be further expanded:

$$\prod_{j=1}^{n} \exp\left(\frac{2\pi i}{q} A_{ij}x_j\right) = \prod_{j,k} \exp\left(\frac{2\pi i}{q} 2^k A_{ij}x_{j,k}\right). \tag{48}$$

[13] We again focus only on the case of the most significant bit, as the k'th most significant bit can be obtained by simply mapping q to $q/2^k$.

Therefore, the desired phase can be applied to the cat state by parallel controlled-Z-rotation gates in constant-quantum depth. Specifically,

$$\left(\prod_{j=1}^{n} \prod_{k=0}^{\lceil \log(q) \rceil - 1} CR_z \left(\frac{2\pi}{q} 2^k A_{i,j} \right) \right) \frac{1}{\sqrt{2}} (|\overline{x_{j,k}}\rangle |\bar{0}\rangle + e^{i\phi_i'(b)} |\overline{x_{j,k}}\rangle |\bar{1}\rangle) = \\ |\overline{x_{j,k}}\rangle \otimes \frac{1}{\sqrt{2}} (|\bar{0}\rangle + e^{i\phi_i(b,x)} |\bar{1}\rangle) \tag{49}$$

where the CR_z gates can be performed in parallel if the size of cat is $\Omega(n \log q) = \Omega(n^2)$. The local quantum circuit for multiplying $A_{i,j}$ with the k'th significant bit of x_j is shown in Figure 5.

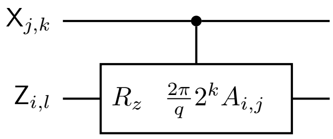

Figure 5: Part of the quantum circuit for matrix-vector multiplication in phase. Here $\mathsf{X}_{j,k}$ is the qubit that stores the k'th bit of x_j, and $\mathsf{Z}_{i,l}$ is the l'th qubit of the cat state storing the information of $|\phi_i\rangle$.

Thus, all operations can be performed in constant quantum depth. □

It is worth noting that in current physical realizations of quantum computers, these (controlled) rotations can be performed directly by tuning microwave frequencies for superconducting qubits [Wen17] or laser frequencies for trapped-ions [BCMS19]. Alternatively, if one insists on having a fixed-size gate set, [HŠ05] provides a constant-depth implementation with $1/poly$ error which is also acceptable.

The Hadamard measurements discussed in the previous sections are performed by measuring X on each qubit of a phase encoded cat state and then taking the parity of the outcomes.

4.3 LWR-based protocol with phase encoding

The protocol using the LWR-based NTCF and the phase encoding is outlined in Figure 6. The verifier behaves essentially the same as in the BCMVV protocol. The major difference is in the prover's honest strategy, which requires it to perform the constant-depth evaluation of the phase encoding.

As we saw in the previous subsections, due to the randomness over the choice of $\mathbf{A}$ and the probabilistic nature of the measurements, the protocol is not perfectly complete. That is, the success probability for the honest prover is no longer 100% as in the original BCMVV protocol. Before accounting for all sources of "imperfections" we first need to examine the post-measurement state in the preimage register after the prover performs step 2 in the protocol. Ideally, we would like this state to be as close as possible to an equal superposition over valid preimages. Thus, in the next subsection we compute a bound on the fidelity of the true state with respect to an ideal state.

4.3.1 Fidelity of the post-measurement state and the success probability for an honest prover

We wish to determine the success probability of an honest prover in the protocol. To do so, we need to characterize the prover's state after it measures the phase-encoded image register. We will show that the state in the preimage register (post-measurement of the phase-encoded image register) has high overlap with the "ideal" preimage state that would have be obtained if the prover performed the evaluation in the computational basis, rather than in phase. With this result, we can then compute the protocol's completeness in the next subsection.

To start the proof we will consider splitting the prover's measurement of the image register into two steps. First, the prover measures in the Hadamard basis all but one qubit from *each* phase encoded state in the image register. Then, it measures the remaining unmeasured qubits as well. This separation is fictitious, as in the protocol the prover will measure all qubits of the

Modified BCMVV protocol

Let $\lambda = n$ be a security parameter and $N \geq 1$ a number of rounds. The parties taking part in the protocol are a PPT machine, known as the verifier and a QPT machine, known as the prover. They will repeat the following steps N times:

1. The verifier generates $(k, t_k) \leftarrow \textsc{Gen}(1^\lambda)$. It sends k to the prover.
2. The prover uses k to implement the phase encoding of the function $g_k(b, x)$, and prepare the following state:
$$|\psi\rangle = \frac{1}{\sqrt{2q^n}} \sum_{b \in \{0,1\}, x \in \mathbb{Z}_q^n} |\bar{b}\rangle_{\mathsf{B}} |\bar{x}\rangle_{\mathsf{X}} |\phi(b,x)\rangle_{\mathsf{Z}}$$
with
$$|\phi(b,x)\rangle = \bigotimes_{i=1}^{n} \bigotimes_{k=1}^{\log_2 p} \left(|\bar{0}\rangle + e^{i\phi_{i,k}} |\bar{1}\rangle\right)^{\otimes v}.$$
where $\phi_{i,k}(b,x) = \frac{2\pi 2^k g_i(b,x)}{q} - \frac{\pi}{2}$. The prover then measures the Z register in Hadamard basis. By conducting majority votes for the parities of the Hadamard measurement outcome of every block $\left(|\bar{0}\rangle + e^{i\phi_{i,k}} |\bar{1}\rangle\right)^{\otimes v}$, the prover obtains a new string $y \in \{0,1\}^{m \log_2 p}$ which it sends to the verifier. The remaining state is
$$|\psi_y\rangle = \sum_{b \in \{0,1\}} |\bar{b}\rangle |\bar{x}_b\rangle |y\rangle. \tag{50}$$
3. The verifier selects a uniformly random challenge $c \leftarrow_R \{0,1\}$ and sends c to the prover.
4. (a) (Preimage test:) When $c = 0$, the prover measures in the standard basis the BX registers of the state leftover in step 2. It obtains the outcomes $b \in \{0,1\}$ and $x \in \{0,1\}^{poly(n)}$, which it sends to the verifier. If $f_k(b,x) = y$, the verifier sets $N_c \leftarrow N_c + 1$.

   (b) (Equation test:) When $c = 1$, the prover measures each qubit in the BX register in the Hadamard basis. It obtains the outcomes $b' \in \{0,1\}^{k'}$ and $d \in \{0,1\}^{poly(n) \cdot k'}$ which it sends to the verifier. Here k' denotes the size of a cat state (used to encode b and each bit in x). The verifier computes $(x_0, x_1) = \text{LWR}\textsc{Inv}(t_k, y)$ and sets $N_c \leftarrow N_c + 1$ if $d \cdot (\bar{x}_0 \oplus \bar{x}_1) = b''$ where b'' is the xor of all k' bits of b'.

At the end of the N rounds, if $\frac{N_c}{N} > 0.95$, the verifier accepts.

Figure 6: Honest provers' strategy for the constant-depth version of the BCMVV protocol [BCM+18] based on phase encoding.

image register in one step. But performing this separation and considering the prover's state after it measures all but one qubit of each phase encoded state will make the analysis simpler. Let us begin with the honest prover's state after performing the coherent evaluation of the function in phase,

$$|\psi\rangle = \frac{1}{\sqrt{2q^n}} \sum_{b \in \{0,1\}} \sum_{x \in \mathbb{Z}_q^n} |\bar{b}, \bar{x}\rangle_{\mathsf{BX}} |\phi(b,x)\rangle_{\mathsf{Z}} \tag{51}$$

where, as before,

$$|\phi(b,x)\rangle = \bigotimes_{i=1}^{n} \bigotimes_{k=1}^{\log_2 p} |\phi_{i,k}(b,x)\rangle^{\otimes v}. \tag{52}$$

Also recall that each component $|\phi_{i,k}\rangle$ has the form of a rotated cat state

$$|\phi_{i,k}(b,x)\rangle = \frac{1}{\sqrt{2}}(|\bar{0}\rangle + e^{i\phi_{i,k}} |\bar{1}\rangle). \tag{53}$$

The prover will measure each qubit of such a state (or, more precisely, of the coherent superposition of such states) in the Hadamard basis. It should be clear that when measuring all but one qubit in the Hadamard basis, the state of that qubit becomes

$$|\tilde{\phi}_{i,k}(b,x)\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm e^{i\phi_{i,k}} |1\rangle), \tag{54}$$

where the ± relative phase is determined by the parity of the Hadamard basis measurement outcomes. Without loss of generality, let us fix[14] this phase as +.

We now rewrite each component $|\tilde{\phi}_{i,k}\rangle$ as

$$\begin{aligned} |\tilde{\phi}_{i,k}(b,x)\rangle =& \frac{1}{\sqrt{2}}(|0\rangle + e^{i\phi_{i,k}}|1\rangle) \\ =& \ \alpha(0|\phi_{i,k})\sqrt{\Pr_{\mathrm{M}}(+|\,|\phi_{i,k}\rangle)}\,|+\rangle + \alpha(1|\phi_{i,k})\sqrt{\Pr_{\mathrm{M}}(-|\,|\phi_{i,k}\rangle)}\,|-\rangle \\ \rightarrow^{H}& \ \alpha(0|\phi_{i,k})\sqrt{\Pr_{\mathrm{M}}(0|\,|\phi_{i,k}\rangle)}\,|0\rangle + \alpha(1|\phi_{i,k})\sqrt{\Pr_{\mathrm{M}}(1|\,|\phi_{i,k}\rangle)}\,|1\rangle \end{aligned} \tag{55}$$

where in the last line we mapped from the Hadamard basis $\{|+\rangle, |-\rangle\}$ to the computational basis $\{|0\rangle, |1\rangle\}$, and $\alpha(0|\phi_{i,k})$ and $\alpha(1|\phi_{i,k})$ are pure phases (i.e. $|\alpha(0|\phi_{i,k})| = |\alpha(1|\phi_{i,k})| = 1$). Let us now consider what happens when all of these qubits are measured. Let $\tilde{z} \in \{0,1\}^{mv\log_2 p}$ denote the Hadamard measurement outcome of all $mv\log_2 p$ $|\tilde{\phi}_{i,k}\rangle$ states. This string can be expressed as a concatenation of $m\log_2 p$ substrings $\tilde{z}_{i,k} \in \{0,1\}^v$ for $i \in \{1,\dots,m\}$ and $k \in \{1,\dots,\log_2 p\}$. The substring with index i,k represents the measurement outcomes of $|\tilde{\phi}_{i,k}\rangle^{\otimes v}$. We can then write the state as

$$\begin{aligned} |\tilde{\phi}_{i,k}(b,x)\rangle^{\otimes v} \rightarrow^{H}& \left(\alpha(0|\phi_{i,k})\sqrt{\Pr_{\mathrm{M}}(0|\,|\phi_{i,k}\rangle)}\,|0\rangle + \alpha(1|\phi_{i,k})\sqrt{\Pr_{\mathrm{M}}(1|\,|\phi_{i,k}\rangle)}\,|1\rangle\right)^{\otimes v} \\ =& \sum_{\tilde{z}_{i,k}\in\{0,1\}^v} \left(\prod_{j=1}^{v} \alpha(\tilde{z}_{i,k,j}|\phi_{i,k})\sqrt{\Pr_{\mathrm{M}}(\tilde{z}_{i,k,j}|\,|\phi_{i,k}\rangle)}\right)|\tilde{z}_i\rangle \\ =& \sum_{\tilde{z}_{i,k}\in\{0,1\}^v} \alpha(\tilde{z}_{i,k}|\phi_{i,k},v)\sqrt{\Pr_{\mathrm{M}}\left(\tilde{z}_{i,k}|\,|\phi_{i,k}\rangle^{\otimes v}\right)}\,|\tilde{z}_i\rangle \end{aligned} \tag{56}$$

where $\tilde{z}_{i,k,j}$ denotes the j'th bit of the substring $\tilde{z}_{i,k}$, and $\alpha(\tilde{z}_{i,k}|\phi_{i,k},v)$ is the product of the pure phases $\alpha(\tilde{z}_{i,k,j}|\phi_{i,k})$ with j ranging from 1 up to v. The entire phase encoded state $|\tilde{\phi}(b,x)\rangle$ can then be expressed as:

$$|\tilde{\phi}(b,x)\rangle \rightarrow^{H} \sum_{\tilde{z}\in\{0,1\}^{mv\log_2 p}} \alpha(\tilde{z}|\phi)\sqrt{\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(b,x)\rangle)}\,|\tilde{z}\rangle\,. \tag{57}$$

Finally, the state of the coherent phase encoding evaluation in Equation 51 (but after the prover has measured all but one qubit of each phase-encoded cat state) can be expressed as well:

$$|\tilde{\psi}\rangle \rightarrow^{H} \frac{1}{\sqrt{2q^n}}\sum_{b,x}|\bar{b},\bar{x}\rangle_{\mathsf{BX}}\sum_{\tilde{z}}\alpha(\tilde{z}|\phi(b,x))\sqrt{\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(b,x)\rangle)}\,|\tilde{z}\rangle_{\mathsf{Z}}\,. \tag{58}$$

Recall that we aim to estimate the success probability of an honest prover. To do so, we can first find an *ideal* state such that, if the prover holds that state, it would very likely succeed in the protocol. The success probability can therefore be estimated by evaluating the fidelity between the real and the ideal states, then evaluating the success probability if the prover holds the ideal state. Denoting the ideal state by $|\psi_{\text{ideal}}\rangle$ and the procedure of majority voting by Maj[15], we let

$$\begin{aligned} |\psi_{\text{ideal}}\rangle =& \frac{c}{\sqrt{2q^n}}\sum_{x_0\in\mathbb{Z}_q^n}\sum_{\mathrm{Maj}(\tilde{z})=f(0,x_0)} \\ & \left(\alpha(\tilde{z}|\phi(0,x_0))\sqrt{\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(\bar{0},\bar{x_0})\rangle)}\,|0,x_0\rangle + \alpha(\tilde{z}|\phi(\bar{1},\bar{x_1}))\sqrt{\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(1,x_1)\rangle)}\,|1,x_1\rangle\right)_{\mathsf{BX}}|\tilde{z}\rangle_{\mathsf{Z}} \end{aligned} \tag{59}$$

[14] We can do this because, as we will see, this is equivalent to the prover having to flip the outcome of one of the measurements it performs. Alternatively, the prover can always perform a quantum-classical interleaving here in order to flip the phase, though this is not necessary.

[15] In other words, $Maj(\tilde{z})$ will be a string of $m\log_2 p$ bits containing the majority value of each substring of v bits.

where c is a normalization constant, x_0 and $x_1 := x_0 - s$ form a claw of $f(b,x)$, hence $f(0,x_0) = f(1,x_1)$. It should be clear why $|\psi_{\mathrm{ideal}}\rangle$ is considered ideal, since the state in the BX register conditioned on having measured Z, will be a superposition of the claw $((0,x_0),(1,x_1))$. This is due to the fact that $\mathrm{Maj}(\tilde{z}) = f(0,x_0)$ which ensures that the image $f(0,x_0)$ can be perfectly decoded. Hence, only the claw $((0,x_0),(1,x_1))$ will be consistent with this outcome of the image register.

We now show the following:

Lemma 4.5. $F(|\tilde{\psi}\rangle, |\psi_{\mathrm{ideal}}\rangle) = |\langle\tilde{\psi}|\psi_{\mathrm{ideal}}\rangle|^2 > 0.98.$

Proof. Let us first give a lower bound of c, where recall that c is the normalization constant in Equation 59. We showed in Theorem 4.2 that at least 99% of $|\phi\rangle$'s are decodable. In other words, we have

$$\sum_{\mathrm{Maj}(\tilde{z})=f(0,x_0)} \Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle) \geq 0.99 \tag{60}$$

and

$$\sum_{\mathrm{Maj}(\tilde{z})=f(1,x_1)} \Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(1,x_1)\rangle) \geq 0.99 \tag{61}$$

for at least 99% possible x_0's. Keeping in mind that $f(0,x_0) = f(1,x_1)$, the normalization condition leads to

$$\frac{c^2}{2q^n} \sum_{x_0 \in \mathbb{Z}_q^n} \sum_{\mathrm{Maj}(\tilde{z})=f(0,x_0)} \left(\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle) + \Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(1,x_1)\rangle)\right) = 1, \tag{62}$$

which implies that

$$1 < c^2 \leq \frac{2q^n}{0.99 \cdot (0.99 + 0.99)q^n} = 1.02. \tag{63}$$

The fidelity can be computed as

$$\begin{aligned} F(|\tilde{\psi}\rangle, |\psi_{\mathrm{ideal}}\rangle) &= |\langle\tilde{\psi}|\psi_{\mathrm{ideal}}\rangle|^2 \\ &\geq \left(\frac{1}{c}\right)^2 > 0.98. \end{aligned} \tag{64}$$

□

In the ideal state, every $\tilde{z}$ measurement outcome corresponds to exactly two $|\phi(b,x)\rangle$ states that form a claw of f. Supposing a specific $\tilde{z}$ is measured, the remaining post-measurement state in the BX register will be

$$|\psi_{\tilde{z}}\rangle \propto \sum_b \alpha(\tilde{z}|\phi(b,x_b)) \sqrt{\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(b,x_b)\rangle)}\, |b,x_b\rangle. \tag{65}$$

Recall that the honest prover would certainly succeed in the protocol with an equal superposition over the claw (without any relative phase between the components):

$$|\psi_y\rangle \propto \sum_b |b,x_b\rangle. \tag{66}$$

Unfortunately, the state in Equation 65, resulting from the measurement of $|\psi_{\mathrm{ideal}}\rangle$, is not of this form due to the presence of the phases $\alpha(\tilde{z}|\phi(\bar{b},\bar{x_b}))$ which could lead to a non-negligible relative phase. We now show that this relative phase is in fact close to zero. To do so, consider a "more ideal state" $|\psi_{\mathrm{ideal},2}\rangle$:

$$\begin{aligned} |\psi_{\mathrm{ideal},2}\rangle = \frac{c'}{\sqrt{2q^n}} &\sum_{x_0 \in \mathbb{Z}_q^n} \sum_{\mathrm{Maj}(\tilde{z})=f(0,x_0)} \\ &\left(\alpha(\tilde{z}|\phi(0,x_0))\sqrt{\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle)}\,|0,x_0\rangle + \alpha(\tilde{z}|\phi(0,x_0))\sqrt{\Pr_{\mathrm{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle)}\,|1,x_1\rangle\right)_{\mathsf{BX}} |\tilde{z}\rangle_{\mathsf{Z}}, \end{aligned} \tag{67}$$

where $c' \in \mathbb{R}$ is another normalization factor. Note that in this state the two components corresponding to the preimage register share the same phase, $\alpha(\tilde{z}|\phi(0,x_0))$, meaning that there is no relative phase.

We start by bounding the normalization constant c' from the norm of the state:

$$1 = \langle\psi_{\text{ideal},2}|\psi_{\text{ideal},2}\rangle = \frac{c'^2}{2q^n}\sum_{x_0\in\mathbb{Z}_q^n}\sum_{\text{Maj}(\tilde{z})=f(0,x_0)} \Pr_{\text{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle)\left(\langle 0,x_0|0,x_0\rangle + \langle 1,x_1|1,x_1\rangle\right)\langle\tilde{z}|\tilde{z}\rangle \tag{68}$$

which implies that

$$1 < c'^2 \leq \frac{2q^n}{0.99q^n\cdot(0.99+0.99)}. \tag{69}$$

It should be clear that if the prover holds $|\psi_{\text{ideal},2}\rangle$, it would succeed in the equation and preimage tests with 100% probability. Thus, to calculate the success probability of the real prover in our protocol, we simply evaluate the fidelity between $|\psi_{\text{ideal}}\rangle$ and $|\psi_{\text{ideal},2}\rangle$.

Lemma 4.6. $F(|\psi_{\text{ideal}}\rangle, |\psi_{\text{ideal},2}\rangle) = |\langle\psi_{\text{ideal}}|\psi_{\text{ideal},2}\rangle|^2 > 0.97$.

Proof.

$$\begin{aligned}|\langle\psi_{\text{ideal}}|\psi_{\text{ideal},2}\rangle| &= \frac{cc'}{2q^n}\sum_{x_0\in\mathbb{Z}_q^n}\sum_{\text{Maj}(\tilde{z})=f(0,x_0)}\\ &\quad\Big[\Pr_{\text{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle) + \alpha^*(\tilde{z}|\phi(1,x_1))\alpha(\tilde{z}|\phi(0,x_0))\sqrt{\Pr_{\text{M}}(\tilde{z}|\,|\phi(1,x_1)\rangle\Pr_{\text{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle}\Big]\\ &\geq \frac{1}{2q^n}\left[0.99q^n\cdot 0.99 + \sum_{x_0}\langle\phi(1,x_1)|\phi(0,x_0)\rangle - 0.01q^n\right],\end{aligned} \tag{70}$$

since

$$\langle\phi(1,x_1)|\phi(0,x_0)\rangle = \sum_{\tilde{z}\in\{0,1\}^{mv\log_2 p}}\alpha^*(\tilde{z}|\phi(1,x_1))\alpha(\tilde{z}|\phi(0,x_0))\sqrt{\Pr_{\text{M}}(\tilde{z}|\,|\phi(1,x_1)\rangle\Pr_{\text{M}}(\tilde{z}|\,|\phi(0,x_0)\rangle}. \tag{71}$$

The inner product $\langle\phi(1,x_1)|\phi(0,x_0)\rangle$ can also be evaluated by considering their phase encoded form. We start with

$$|\phi(b,x)\rangle = \bigotimes_{i=1}^{m}\bigotimes_{k=1}^{\log_2 p}\left(|\bar{0}\rangle + e^{i\phi_{i,k}}|\bar{1}\rangle\right)^{\otimes v}$$

where $\phi_{i,k}(b,x) = \frac{2^k\pi g_i(b,x)}{q} - \frac{\pi}{2}$. As both states are phase encodings, the inner product will be determined by the angle differences between the components. In other words, letting

$$\Delta\phi_{i,k} = \frac{2^k\pi(g_i(0,x) - g_i(1,x-s))}{q} \tag{72}$$

and noting that $g_i(0,x) = (\mathbf{A}x)_i$ and $g_i(1,x-s) = (\mathbf{A}x)_i + e_i$, it is the case that

$$\Delta\phi_{i,k} = \frac{2^k\pi e_i}{q}. \tag{73}$$

We can now express the inner product as

$$\begin{aligned}\langle\phi(0,x)|\phi(1,x-s)\rangle &= \prod_{i,k}\left(\exp\left(i\frac{\Delta\phi_{i,k}}{2}\right)\cos\left(\frac{\Delta\phi_{i,k}}{2}\right)\right)^v\\ &= \exp\left(i\sum_{i,k}\frac{\Delta\phi_{i,k}\cdot v}{2}\right)\prod_{i,k}\left(\cos\left(\frac{\Delta\phi_{i,k}}{2}\right)\right)^v\end{aligned} \tag{74}$$

But now note that $\|e\|_\infty \leq \frac{cq}{p^5}$, for some constant $c > 0$, as per Definition 4.1. If we substitute this into the formula for $\Delta\phi_{i,k}$, keeping in mind that $2^k \leq p$, we find that

$$\Delta\phi_{i,k} = \frac{2^k \pi e_i}{q} \leq \frac{\pi}{p^4}. \tag{75}$$

Taking n to be sufficiently large, so that p is sufficiently large, leads to

$$\cos\left(\frac{\Delta\phi_{i,k}}{2}\right) \geq 1 - \frac{\pi^2}{8p^8} - O\left(p^{-16}\right) \tag{76}$$

and

$$\prod_{i,k}\cos\left(\frac{\Delta\phi_{i,k}}{2}\right)^v \geq \left(1 - \frac{\pi^2}{8p^8}\right)^{mv\log_2 p}. \tag{77}$$

But now $p^8 = O\left((mn\log q)^4\right) = O(n^{16})$ and $mv\log_2 p = O(n^2 \cdot n^4 \log n \cdot \log n) = O(n^6 \log^2 n)$. It follows that

$$\prod_{i,k}\cos\left(\frac{\Delta\phi_{i,k}}{2}\right)^v \geq 1 - \frac{1}{poly(n)}. \tag{78}$$

For the phase part

$$\sum_{i,k}\frac{\Delta\phi_{i,k}\cdot v}{2} = O\left(p^{-4}(m\log_2 p)^3\log_2(m\log_2 p)\right) = O\left(\frac{(\log_2 n)^4}{n^2}\right) \tag{79}$$

and similarly

$$\exp\left(i\sum_{i,k}\frac{\Delta\phi_{i,k}\cdot v}{2}\right) = 1 - O\left(\frac{(\log_2 n)^8}{n^4}\right) + iO\left(\frac{(\log_2 n)^4}{n^2}\right) = 1 - \frac{1}{poly(n)} + i\cdot\frac{1}{poly(n)}. \tag{80}$$

Finally,

$$\langle\phi(0,x)|\phi(1,x-s)\rangle = 1 - \frac{1}{poly(n)} + i\cdot\frac{1}{poly(n)} \tag{81}$$

and the fidelity can be lower-bounded as follows

$$\begin{aligned} |\langle\psi_{\text{ideal}}|\psi_{\text{ideal},2}\rangle|^2 &\geq \left[\frac{1}{2q^n}\left(0.99q^n\cdot 0.99 + \sum_{x_0}\left(1 - \frac{1}{poly(n)} + i\cdot\frac{1}{poly(n)}\right) - 0.01q^n\right)\right]^2 \\ &= \left(-\frac{0.01}{2} + \frac{1}{2}\left|0.98 + 1 - \frac{1}{poly(n)} + i\cdot\frac{1}{poly(n)}\right|\right)^2 \\ &= \left[-\frac{0.01}{2} + \frac{1}{2}\sqrt{\left(1.98 - \frac{1}{poly(n)}\right)^2 + \left(\frac{1}{poly(n)}\right)^2}\right]^2 \\ &= \left[\frac{-0.01 + 1.98 - \frac{1}{poly(n)}}{2}\right]^2 = \left[\frac{1.97 - \frac{1}{poly(n)}}{2}\right]^2 > 0.97 \end{aligned} \tag{82}$$

for large sufficiently n. □

Combining Lemmas 4.5 and 4.6, we conclude that the success probability for an honest prover is lower bounded by 0.95, using a union bound.

4.3.2 Completeness

We can now compute the probability for an honest prover, following the strategy outlined in Figure 6, to pass the verifier's checks. We start with the observation that q is prime. As mentioned, this would require the prover to create a superposition in the preimage register of q^n components. Instead, the prover creates a superposition of q'^n components, where q' is a power of 2 that is close to q. From the results in Subsection 4.1.1, we incur a $O(n^{-1})$ penalty in the honest prover's success probability as a result of this. Next, we saw that when performing the measurement of the image register, there is a chance that the $|\phi(b,x)\rangle$ state contains components that are undecodable. We limited the probability of this happening to 1%, with the parameter choices mentioned in Subsection 4.2.2. Assuming the state is decodable, we saw that the probability of incorrectly decoding is also 1%. With these results, we showed in Subsection 4.3.1 that the prover's state, upon measuring the image register (and successfully decoding the result, which is sent to the verifier), gives it at least a 95% success probability in the equation and preimage tests. This also accounted for the failure probability of incorrectly decoding the image register. Finally, as discussed in Subsection 4.2.4, if we choose to use a fixed-size gate set, we will incur another $1/poly(n)$ error.

Putting everything together, we find that the overall completeness of the protocol is $95\% - O(n^{-1})$.

4.3.3 Soundness

Since we showed that the LWR-based function $f(b,x)$ is also an NTCF, in Subsection 4.1, our new constant quantum depth protocol inherits the soundness of the original BCMVV protocol.

4.4 Resource estimation

As in Subsection 3.3, we summarize the resources required for an honest prover to succeed in the protocol.

4.4.1 Quantum depth and quantum-classical interleavings

1. **Preparation of cat states.** Same as in the randomized encoding construction, the depth of this step is 5 and the prover interleaves constant-depth quantum computation and classical log-depth computation once.

2. **Evaluation of the LWR function by phase encoding.** As is illustrated in Figure 5, this step consists of only parallel CR_z gates or $\mathrm{R}_z(\frac{\pi}{2})$ gates. The depth added is only 1 for the example case in Figure 5.

3. **Measurement of the Z register.** As is explained in Subsection 4.2.2, the measurement of the Z register contains Hadamard measurements and a majority vote (performed classically on the measurement outcome), hence this step has quantum depth 2 and adds 1 step of quantum-classical interleaving.

4. **Preimage test/equation test.** Exactly the same as in the BCMVV protocol, this step requires at most depth 2 and 1 interleaving for the equation test.

In summary, the phase encoding construction requires even shorter quantum depth than the generic construction, as the overall quantum depth is $5+1+2+2=10$. The number of quantum-classical interleaving is 3, same as the generic construction.

4.4.2 Circuit width

The total width of the circuit is determined by the product of several multipliers in the protocol:

1. **Number of output components of** $g(b,x)$ is $O(m)=O(n^2)$, by definition.

2. $\lfloor\cdot\rfloor_p$ **rounding function.** The phase encoding needs to be prepared for all of the $\log p$ bits. This leads to another $O(\log(\sqrt{mn\log q}))=O(\log m)=O(\log n)$ multiplier.

3. **Cat state.** As discussed in Subsection 4.2.4, the size of the cat state for each component $|\phi_i\rangle$ needs to be $O(n \log q) = O(n^2)$.

4. **Repetition for majority votes.** This is calculated in Subsection 4.2.3 and each $|\phi_i\rangle$ needs to be repeated for $v = O(n^4 \log^2 n)$ times.

In summary, the total circuit width required is $O(n^8 \log^3 n)$. Although this is still a high-order polynomial, it is a significant improvement over the randomized encoding construction (where we estimated $O(n^{33})$ width). Note that the normal, poly-depth, construction requires $O(m \log q) = O(n^3)$ width.

It is also worth mentioning that there can be a trade-off between the size of the cat states and the depth of the circuit, since the matrix multiplication does not need to be fully parallelized. In practice, one can double the number of CR_z gates applied on each qubit to halve the width.

4.5 Robustness against noise

Another feature of our phase encoding construction is some amount of intrinsic robustness against noise, which makes it closer to practical use on near-term devices.

The key reasons for the noise-resistance are the use of cat states, the classical repetition code we applied in measuring the Z register, as is discussed in Subsection 4.2.2, the error-correcting properties of the LWR construction which we used implicitly in Subsection 4.3.1 and the constant gap between the best quantum strategy and the best classical strategy (assuming intractability of LWE) as encapsulated by Inequality 17.

We can therefore see that errors on the image register, Z, may lead to bit flips of the output string z such that $z \neq y$ (where recall that y is the ideal decoding). However, since any bit z_i is determined by majority voting for all v repetitions of the phase encoding of that bit, the probability that z_i is flipped is much smaller than that of single bit flipping. Intuitively speaking, some correctly measured bits may be flipped due to noise that might appear in any stage of the protocol, but incorrect bits are equally likely to be flipped. Hence the majority vote will still very likely output $z_i = y_i$.

A repetition code is also used indirectly in the preimage register, as the preimages are encoded in cat states. While this makes the preimage test robust to noise, the equation test will not be, in general. This is because in the equation test, the prover needs to report a string d and a bit b such that

$$d \cdot (\bar{x}_0 \oplus \bar{x}_1) = b \tag{83}$$

where $\bar{x}_0$ and $\bar{x}_1$ are the repetition code encodings of preimages x_0 and x_1 (that match the image the prover returned in the previous round of the protocol). In this case we can see that even a single bit flip in either the string d or of the bit b can make the equation invalid. We therefore leave it as an open problem to find a fully noise-robust implementation of the protocol.

References

[AA11] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 333–342, 2011.

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[AAMA+21] MD SAJID ANIS, Abby-Mitchell, Héctor Abraham, AduOffei, et al. Qiskit: An open-source framework for quantum computing, 2021.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[AC17] Scott Aaronson and Lijie Chen. Complexity-Theoretic Foundations of Quantum Supremacy Experiments. In *Proceedings of the 32nd Computational Complexity Conference*, CCC '17, pages 1–67, Dagstuhl, DEU, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[AG20] Scott Aaronson and Sam Gunn. On the Classical Hardness of Spoofing Linear Cross-Entropy Benchmarking. *Theory of Computing*, 16(11):1–8, 2020.

[AIK04] B. Applebaum, Y. Ishai, and E. Kushilevitz. Cryptography in NC^0. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 166–175, 2004.

[AKPW13] Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with Rounding, Revisited. In *Advances in Cryptology – CRYPTO 2013*, pages 57–74, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Bar89] David A Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

[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.

[BCMS19] Colin D. Bruzewicz, John Chiaverini, Robert McConnell, and Jeremy M. Sage. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 2019.

[BFNV19] Adam Bouland, Bill Fefferman, Chinmay Nirkhe, and Umesh Vazirani. On the complexity and verification of quantum random circuit sampling. *Nature Physics*, 15(2):159–163, Feb 2019.

[BIS$^+$18] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, Michael J Bremner, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *Nature Physics*, 14(6):595–600, 2018.

[BKVV20] Zvika Brakerski, Venkata Koppula, Umesh Vazirani, and Thomas Vidick. Simpler Proofs of Quantumness. In *15th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2020)*, volume 158 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom Functions and Lattices. In *Advances in Cryptology – EUROCRYPT 2012*, pages 719–737. Springer Berlin Heidelberg, 2012.

[CHSH69] John F Clauser, Michael A Horne, Abner Shimony, and Richard A Holt. Proposed experiment to test local hidden-variable theories. *Physical Review Letters*, 23(15):880, 1969.

[CSV21] Matthew Coudron, Jalex Stark, and Thomas Vidick. Trading locality for time: certifiable randomness from low-depth circuits. *Communications in Mathematical Physics*, 382(1):49–86, 2021.

[CW00] Richard Cleve and John Watrous. Fast parallel circuits for the quantum Fourier transform. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 526–536. IEEE, 2000.

[Dus98] Pierre Dusart. *Autour de la fonction qui compte le nombre de nombres premiers*. PhD thesis, Université de Limoges, 1998.

[FMMC12] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[Gal22] François Le Gall. Private correspondence, 2022.

[GE21] Craig Gidney and Martin Ekerå. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021.

[GH20] Alexandru Gheorghiu and Matty J Hoban. Estimating the entropy of shallow circuit outputs is hard. *arXiv preprint arXiv:2002.12814*, 2020.

[HG21] Shuichi Hirahara and François Le Gall. Test of Quantumness with Small-Depth Quantum Circuits. In *46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021)*, volume 202 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 59:1–59:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[HM17] Aram W Harrow and Ashley Montanaro. Quantum computational supremacy. *Nature*, 549(7671):203–209, 2017.

[HŠ05] Peter Høyer and Robert Špalek. Quantum Fan-out is Powerful. *Theory of Computing*, 1(5):81–103, 2005.

[HZN+20] Cupjin Huang, Fang Zhang, Michael Newman, Junjie Cai, Xun Gao, Zhengxiong Tian, Junyin Wu, Haihong Xu, Huanjun Yu, Bo Yuan, Mario Szegedy, Yaoyun Shi, and Jianxin Chen. Classical Simulation of Quantum Supremacy Circuits. *arXiv preprint arXiv:2005.06787*, 2020.

[KM19] Gregory D Kahanamoku-Meyer. Forging quantum data: classically defeating an IQP-based quantum test. *arXiv preprint arXiv:1912.05547*, 2019.

[KMCVY22] Gregory D. Kahanamoku-Meyer, Soonwon Choi, Umesh V. Vazirani, and Norman Y. Yao. Classically verifiable quantum advantage from a computational Bell test. *Nature Physics*, 18(8):918–924, 2022.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–23. Springer, 2010.

[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.

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

[PR22] A. S. Popova and A.N. Rubtsov. Cracking the Quantum Advantage Threshold for Gaussian Boson Sampling. In *Quantum 2.0 Conference and Exhibition*, page QW2A.15. Optica Publishing Group, 2022.

[Pre18] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[Rab80] Michael O Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138, 1980.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):1–40, 2009.

[SB09] Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 465(2105):1413–1439, 2009.

[Sho94] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. IEEE, 1994.

[WBC$^+$21] Yulin Wu, Wan-Su Bao, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Liping Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Qingling Zhu, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. Strong Quantum Computational Advantage Using a Superconducting Quantum Processor. *Phys. Rev. Lett.*, 127:180501, 2021.

[WBD$^+$19] K Wright, KM Beck, Sea Debnath, JM Amini, Y Nam, N Grzesiak, J-S Chen, NC Pisenti, M Chmielewski, C Collins, et al. Benchmarking an 11-qubit quantum computer. *Nature communications*, 10(1):1–6, 2019.

[Wen17] G Wendin. Quantum information processing with superconducting circuits: a review. *Reports on Progress in Physics*, 80(10):106001, 2017.

[WKST19] Adam Bene Watts, Robin Kothari, Luke Schaeffer, and Avishay Tal. Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 515–526, 2019.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167. IEEE, 1986.

[ZCC$^+$22] Qingling Zhu, Sirui Cao, Fusheng Chen, Ming-Cheng Chen, Xiawei Chen, Tung-Hsun Chung, Hui Deng, Yajie Du, Daojin Fan, Ming Gong, Cheng Guo, Chu Guo, Shaojun Guo, Lianchen Han, Linyin Hong, He-Liang Huang, Yong-Heng Huo, Liping Li, Na Li, Shaowei Li, Yuan Li, Futian Liang, Chun Lin, Jin Lin, Haoran Qian, Dan Qiao, Hao Rong, Hong Su, Lihua Sun, Liangyuan Wang, Shiyu Wang, Dachao Wu, Yulin Wu, Yu Xu, Kai Yan, Weifeng Yang, Yang Yang, Yangsen Ye, Jianghan Yin, Chong Ying, Jiale Yu, Chen Zha, Cha Zhang, Haibin Zhang, Kaili Zhang, Yiming Zhang, Han Zhao, Youwei Zhao, Liang Zhou, Chao-Yang Lu, Cheng-Zhi Peng, Xiaobo Zhu, and Jian-Wei Pan. Quantum computational advantage via 60-qubit 24-cycle random circuit sampling. *Science Bulletin*, 67(3):240–245, 2022.

[ZKML$^+$21] Daiwei Zhu, Gregory D. Kahanamoku-Meyer, Laura Lewis, Crystal Noel, Or Katz, Bahaa Harraz, Qingfeng Wang, Andrew Risinger, Lei Feng, Debopriyo Biswas, Laird Egan, Alexandru Gheorghiu, Yunseong Nam, Thomas Vidick, Umesh Vazirani, Norman Y. Yao, Marko Cetina, and Christopher Monroe. Interactive Protocols for Classically-Verifiable Quantum Advantage. *arXiv preprint arXiv:2112.05156*, 2021.

[ZWD$^+$20] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, Peng Hu, Xiao-Yan Yang, Wei-Jun Zhang, Hao Li, Yuxuan Li, Xiao Jiang, Lin Gan, Guangwen Yang, Lixing You, Zhen Wang, Li Li, Nai-Le Liu, Chao-Yang Lu, and Jian-Wei Pan. Quantum computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

A Randomized encoding construction from [AIK04]

The construction of randomized encodings from [AIK04] is based on *branching programs*. We are only interested in mod-2 branching programs, which we define here:

Definition A.1 (Branching programs [AIK04])**.** A branching program (BP) is defined by a tuple $BP = (G, \phi, s, t)$ where $G = (V, E)$ is a directed acyclic graph, ϕ is a labeling function assigning each edge either a positive literal x_i or a negative literal $\neg x_i$. An input binary vector $\vec{w}$ determines a subgraph G_w where an edge labeled as x_i is preserved if and only if $w_i = 1$. In a (counting) mod-2 BP, the BP computes the number of paths from s to t modulo 2. The size, l, of a BP is defined as the number of vertices, $|V|$.

As an example, Figure 7 shows a mod-2 branching program of size $l = 4$ and having three inputs $x = (x_0, x_1, x_2)$.

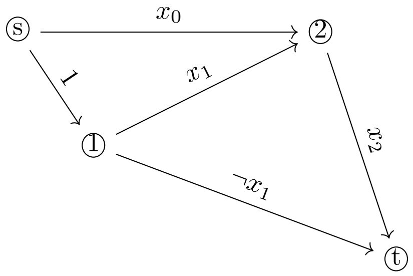

Figure 7: This size-4 mod-2 branching program consists of 5 edges whose connectivity is decided by the value of the input bits. Note that $\neg x_1$ means that this edge is available if and only if $x_1 = 0$. As an example, when the input $x = (x_0, x_1, x_2) = (0, 1, 1)$, there is only one path from s to t which is ⓢ − ① − ② − ⓣ. Thus the output of this mod-2 BP will be 1.

We now state one of the most important results concerning branching programs, due to Barrington:

Theorem A.1 (Barrington's theorem [Bar89])**.** If $f : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit of depth d, then it can be computed by a branching program of width 5 and length $O(4^d)$.

The above theorem ensures that the log-depth (N)TCFs used in proof of quantumness protocols can be transformed into polynomial-size branching program. Given that branching programs output a single bit, this construction has to be performed for every output bit of a (N)TCF.

A size-l mod-2 BP for a binary function f can be represented by an adjacency matrix since BPs are directed acyclic graphs. Let $A(x)$ denote the $l \times l$ adjacency matrix of a BP with input x. We also denote as $L(x)$ the $(l-1) \times (l-1)$ submatrix of $A(x) - I$ obtained by deleting the first column and the last row. It turns out that the following fact holds:

Lemma A.1 ([AIK04])**.** $f(x) = \det(L(x))$ *mod* 2.

This lemma is the basis for constructing a randomized encoding for f. The goal will be to "garble" $L(x)$ through products with certain random matrices. The garbling should be done in such a way that the determinant of the resulting matrix matches that of $L(x)$, thus preserving the correctness of the construction.

To that end, let $r^{(1)} \leftarrow_R \{0,1\}^{\binom{l-1}{2}}$ and $r^{(2)} \leftarrow_R \{0,1\}^{l-2}$. Use these to construct matrices $R^{(1)}$ and $R^{(2)}$ of dimensions $(l-1) \times (l-1)$. Both matrices have all diagonal elements equal to 1. The right upper-diagonal elements of $R^{(1)}$ (that is, the entries $R^{(1)}_{i,j}$ with $j > i$) are filled with the entries of $r^{(1)}$. The last column of $R^{(2)}$, except for the last element, (that is, the entries $R^{(2)}_{i,l-1}$, $1 \leq i \leq l-2$) is filled with the elements of $r^{(2)}$. All other entries of $R^{(1)}$ and $R^{(2)}$ are 0. The following can be shown:

Lemma A.2 ([AIK04])**.** $\det(L(x)) = \det(R^{(1)} L(x) R^{(2)})$

This is not too difficult to see, as both $R^{(1)}$ and $R^{(2)}$ have determinant 1. One now defines the randomized encoding $\tilde{f}(x, r^{(1)}, r^{(2)}) = R^{(1)} L(x) R^{(2)}$. It follows that:

Lemma A.3 ([AIK04])**.** $\tilde{f}$ is a perfect randomized encoding of f.

By construction, every entry of $\tilde{f}$ is a degree-3 polynomial in its input variables. However, computing this function (i.e. computing every matrix entry of $R^{(1)}L(x)R^{(2)}$) cannot be done in constant-depth. The reason is that some of the input variables are involved in a linear number of monomials of the output. To compute the function in constant depth, it must be that each input variable appears in only a constant number of monomials. The authors of [AIK04] remedy this by considering a randomized encoding for $\tilde{f}$. Before doing so, note that

Lemma A.4 ([AIK04])**.** The composition of perfect randomized encodings is still a perfect randomized encoding of the original function.

Thus, a randomized encoding for $\tilde{f}$ will also be a randomized encoding for f. Denote the i, j entry of $\tilde{f}$ as $\tilde{f}_{i,j}$. We can see that

$$\tilde{f}_{i,j}(x, r^{(1)}, r^{(2)}) = T_1(x, r^{(1)}, r^{(2)}) \oplus T_2(x, r^{(1)}, r^{(2)}) \oplus ... \oplus T_k(x, r^{(1)}, r^{(2)}) \tag{84}$$

where each T_m is a monomial in the input variables. Finally, define $\hat{f}$ as

$$\hat{f}_{i,j}(x, r^{(1)}, r^{(2)}, r, r') = (T_1 \oplus r_1, T_2 \oplus r_2, ..., T_k \oplus r_k, r_1 \oplus r'_1, r'_1 \oplus r_2 \oplus r'_2, ..., r'_{k-1} \oplus r_k) \tag{85}$$

where r and r' are newly introduced vectors of random bits. Note that adding all entries in 85 results in the summation from Equation 84. Thus, $\hat{f}$ contains all of the information required to compute $\tilde{f}$ and moreover,

Lemma A.5 ([AIK04])**.** $\hat{f}$ is a perfect randomized encoding of f with output locality 4.

Here, output locality 4 means that each output bit depends on at most 4 input bits, which immediately implies that the function can be evaluated in constant depth. The classical circuit computing an entry of $\hat{f}$ is shown in Figure 8. Detailed proofs of all these results can be found in [AIK04].

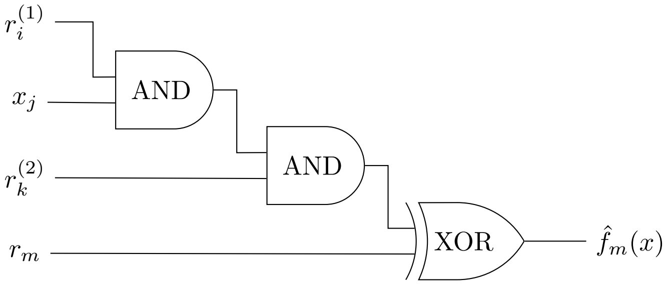

Figure 8: The circuit for evaluating each entry in the randomized encoding $\hat{f}$. The circuit shown here computes the m'th entry, with $m \leq k$, consisting of the monomial $r_i^{(1)} x_j r_k^{(2)}$ xored with r_m. For the entries with $m > k$, note that a single XOR gate is required.

B Reconstruction of randomness

In our first constant quantum-depth proof of quantumness, the prover is instructed to evaluate a randomized encoding of a TCF. The verifier must still be able to use the trapdoor in order to invert an output of the randomized encoding. As mentioned in Subsection 3.2, this is true provided the encoding satisfies the randomness reconstruction property. Here we prove this fact for the construction of [AIK04].

Proof of Lemma 2.2. We would like to show that given an instance of $\hat{f}_{i,j}(x, r^{(1)}, r^{(2)}, r, r')$, as shown in Equation 85, as well as x, it is possible to efficiently recover the randomness $r^{(1)}, r^{(2)}, r, r'$. First note that if the terms T_k were known as well as $r^{(1)}, r^{(2)}$, it is straightforward to recover r and r'. We will therefore focus on that case. From Equation 85 it is possible to efficiently compute the result of Equation 84, since $\hat{f}$ is a randomized encoding of $\tilde{f}$: simply xor all the terms in

Equation 85. We will then focus on randomness reconstruction for $\bar{f}$ as that will then yield randomness reconstruction for $\hat{f}$.

Denote as $M = \tilde{f}(x, r^{(1)}, r^{(2)}) = R^{(1)}L(x)R^{(2)}$. Given M and x we wish to recover $r^{(1)}, r^{(2)}$. This boils down to solving a specific quadratic system of equations. To see why, take $l = 4$ as an example,

$$M = R^{(1)}L(x)R^{(2)} = \begin{bmatrix} 1 & r_1^{(1)} & r_3^{(1)} \\ 0 & 1 & r_2^{(1)} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_4 & x_6 \\ -1 & x_2 & x_5 \\ 0 & -1 & x_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & r_1^{(2)} \\ 0 & 1 & r_2^{(2)} \\ 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} x_1 - r_1^{(1)} & r_1^{(1)}x_2 - r_3^{(1)} + x_4 & r_1^{(2)}(x_1 - r_1^{(1)}) + r_2^{(2)}(r_1^{(1)}x_2 - r_3^{(1)} + x_4) + r_3^{(1)}x_3 + r_1^{(1)}x_5 + x_6 \\ -1 & x_2 - r_2^{(1)} & r_2^{(2)}(x_2 - r_2^{(1)}) + r_2^{(1)}x_3 - r_1^{(2)} + x_5 \\ 0 & -1 & x_3 - r_2^{(2)} \end{bmatrix}$$

Note that the main diagonal of M is just a linear system of 3 equations with 3 unknowns. It can therefore be solved, yielding $r_1^{(1)}$, $r_2^{(1)}$ and $r_2^{(2)}$. Plugging these values into the second diagonal (the one above the main diagonal), yields another system of linear equations with an equal number of unknowns. By repeating the process and solving all of these systems, all bits in $r^{(1)}$ and $r^{(2)}$ are recovered.

We now show that this strategy works for arbitrary l. Start by observing that:

$$\begin{cases} R_{i,j}^{(1)} = 1, & i = j \\ R_{i,j}^{(1)} = 0, & i > j, \end{cases}$$

$$\begin{array}{ll} L_{i,j} = -1, & i = j+1 \\ L_{i,j} = 0, & i > j+1, \end{array}$$

$$\begin{cases} R_{i,j}^{(2)} = 1, & i = j \\ R_{i,j}^{(2)} = 0, & (i > j) \vee (i < j < l-2). \end{cases}$$

The entries of M can then be expressed as:

$$M_{i,j} = \sum_{k_1,k_2} R_{i,k_1}^{(1)} L_{k_1,k_2} R_{k_2,j}^{(2)}.$$

Consider the entries on the main diagonal, excluding the last element:

$$M_{i,i} = \sum_{k_1} R_{i,k_1}^{(1)} L_{k_1,i} = R_{i,i}^{(1)} L_{i,i} + \sum_{k_1 > i} R_{i,k_1}^{(1)} L_{k_1,i} = L_{i,i} - R_{i,i+1}^{(1)}$$

with $i < l-2$ and where $R_{i,i+1}^{(1)}$ are the elements of the second diagonal of $R^{(1)}$ and the $L_{i,i}$'s are already known (as they only involve entries of x). This gives us a simple linear system which we can solve to recover the $R_{i,i+1}^{(1)}$ values. Then, for $i = l-2$:

$$M_{l-2,l-2} = \sum_{k_2} L_{l-2,k_2} R_{k_2,l-2}^{(2)} = L_{l-2,l-3} R_{l-3,l-2}^{(2)} + L_{l-2,l-2} R_{l-2,l-2}^{(2)} = R_{l-3,l-2}^{(2)} + L_{l-2,l-2}.$$

From this we also recover $R_{l-3,l-2}^{(2)}$, i.e. the last entry in $r^{(2)}$. Note that the unknowns here consisted of the entries in the second diagonal of $R^{(1)}$ and the last element of $r^{(2)}$. This matches the number of equations and so all values could be recovered.

We now claim that the k'th diagonal of M is a linear system which depends *only* on the $k+1$ diagonal of $R^{(1)}$ and the k'th last element of $r^{(2)}$ given the solutions to the previous $k-1$ diagonals of M. Writing out the elements, we have:

$$M_{i,i+j} = \sum_{k_1,k_2} R_{i,k_1}^{(1)} L_{k_1,k_2} R_{k_2,i+j}^{(2)}.$$

with $j = k - 1$. For $i + j \neq l - 2$:

$$M_{i,i+j} = \sum_{k_1} R^{(1)}_{i,k_1} L_{k_1,i+j} = -R^{(1)}_{i,i+j+1} + L_{i,i+j} + \sum_{i<k_1<i+j+1} R^{(1)}_{i,k_1} L_{k_1,i+j}$$

where the first term is from the $(k+1)$'th diagonal of $R^{(1)}$ and the remaining terms are known from solving the equations for the previous diagonals. Thus, we have a linear system, which we can solve, with unknowns comprising the elements of the $(k+1)$'th diagonal of $R^{(1)}$.

For $i + j = l - 2$:

$$M_{l-2-j,l-2} = \sum_{k_2=l-2-j-1}^{l-2} L_{l-2-j,k_2} R^{(2)}_{k_2,l-2} + \sum_{k_1=l-2-j+1}^{l-2} R^{(1)}_{l-2-j,k_1} \sum_{k_2=k_1-1}^{l-2} L_{k_1,k_2} R^{(2)}_{k_2,l-2}.$$

The first term is a linear combination of the last $k+1$ entries of $R^{(2)}$, i.e. the last k elements of $r^{(2)}$, and only the k'th element is unknown. The remaining terms are known from solving the systems corresponding to the previous diagonals.

We can therefore proceed in this fashion, starting from the first diagonal of M and going upwards solving all systems of linear equations and thus recovering all values of $r^{(1)}$ and $r^{(2)}$. This procedure is clearly efficient and we have shown that it is also correct. To conclude the proof, we also need to make sure that there is a unique solution to the system. This is guaranteed by the unique randomness property of the randomized encoding (Theorem 2.5). □