Quantum Random Number Generation

Henrique Assumpção* and Diogo Neiss†

Computer Science Department

Universidade Federal de Minas Gerais

November 3, 2023

The ability to reliably generate random numbers is essential to most professionals in STEM, specially for those working with statistical and computational experiments. Most RNG systems today, however, generate pseudorandom numbers that are not truly random in nature, and although this approximation may suffice for most applications, there are a myriad of relevant problems that require a way of generating true random numbers, such as high-security networks and proof-of-stake cryptocurrencies. In this work, we present a comprehensive review of one of the most important methods for generating true random numbers: Quantum Random Number Generation. First, we introduce some preliminary concepts that are paramount to fully understanding the topic at hand, and then we perform an in-depth look the concept of Quantum Randomness and its uses to random number generation. Second, we discuss a novel method for generating cryptographically certified random bits based on random circuit sampling protocols, that was recently proposed by Scott Aaronson and Shih-Han Hung.

Keywords: Quantum Random Number Generation, Quantum Supremacy, Quantum Computing

*henriquesoares@dcc.ufmg.br
†diogoneiss@dcc.ufmg.br

Contents

1 Introduction

Many different areas in Computer Science, Physics, Chemistry and Engineering have a deep reliance on random numbers to perform a myriad of different tasks. In computer science alone problems from cryptography to artificial intelligence depend on the ability to efficiently and reliably generate random numbers. However, as there is no way to achieve fast and reliable true randomness in a deterministic system, where correlations appear on large scale generation, one must resort to physical sources of entropy, which in turn are often troubled by low throughput of entropy coupled with uncertainty regarding the hardware's reliability.

The concept of "certifiable" randomness adds another crucial layer to this discussion. Certifiable random numbers are those whose randomness can be mathematically verified. As reported previously in the media, the NSA *Bullrun* operation inserted backdoors into some mainstream algorithms, rendering their unpredictable falsified. With the rapid advancement of technology and the ever-increasing need for secure data transmission, it's increasingly necessary to be able to guarantee the integrity of random numbers.

Quantum Random Number Generation then appears as an ideal candidate for solving the problem of generating random numbers, as quantum mechanics is inherently stochastic — you don't need to source entropy across time, true randomness is a fundamental reality –, thus serving as an ideal source for random number generation. Moreover, quantum processes, coupled with the principles of quantum cryptography, can allow us to generate cryptographically certified random bits, which in turn may make our systems more robust and secure.

The main result discussed in this paper concerns precisely that: how to generate random numbers that are certifiably random using a quantum computer. A recent result by Scott Aaronson and Shih-Han Hung proposes a quantum RNG system that utilizes the recent Quantum Supremacy experiments – conducted by Google in 2019 – to design a system that can be mathematically proven to output a minimum amount of entropy, given certain reasonable computational assumptions.

The main objective of this article then is as follows:

- We first introduce the concept of Quantum Random Number Generation, and formalize the many different ways to harness the inherent randomness of quantum systems.
- We then introduce and formally prove the main results from Aaronson and Hung's [AH23] for generating cryptographically certified random bits.

2 Preliminaries

In this section, we formalize the main results that we consider to be essential prerequisites for understanding further sections of this work. However, the descriptions in this section will be brief, and thus we are assuming that the reader is familiar with most of the discussed topics.

The main references for the next subsections are as follows: subsection 2.1 uses [Khr15, Kol63, Kol98, Cha69], subsection 2.2 uses [Mac03], subsection 2.3 uses [L'E17, Knu97], subsection 2.4 uses [NC00, AH23, Wik23c], and subsection 2.5 uses [NC00, Wik23a, Wik23b, Bab85].

2.1 Randomness

First, we discuss the meaning of being *random* and how this concept is related to probability theory.

Consider a hypothetical set, denoted as Ω, consisting of all sequences of length N with entries in $\{0,1\}^n$. A sample of such a sequence would be $x = (x_1, ..., x_N)$, where each x_j could either be 0 or 1. We can assign a uniform probability distribution to Ω, meaning that each sequence has an equal probability of occurring, calculated as $p(x) = \frac{1}{2^N}$ for each $x \in \Omega$.

Now, imagine that, after an experiment is conducted, the following relatively long sample is obtained:

$$
\begin{aligned}
&100100100001111110110101010001000100001011010001100001000110100 1\\
&100010011000110011000101000101110000000110111000001110011010001 0\\
&101001000000100100111000001000100010100110011111100110001110100 00\\
&000010000010111011111010100110001110110001001110011010110010001001
\end{aligned}
$$

This sequence *seems* to fit the intuitive understanding of randomness, as it does not seem to hold an evident pattern. On the other hand, if we consider a different sequence of the same length, say one that alternates between 0 and 1, $(01)^n$, such that n is the number of repetitions, we notice that, even though this sequence is equally as likely to occur as the previous string, we wouldn't typically classify it as 'random' due to its obvious pattern. If such an alternating sequence were to appear in an experiment intended to produce random sequences, it would be surprising and inconsistent with intuitive expectations of randomness, even though it is still possible. Yet, traditional probability calculations don't capture this notion of *uncertain unique outcomes*, illustrating the limits of probability theory when it comes to capturing our intuitive understanding of randomness.

Our "random" long sequence was actually taken from the first 258 binary digits of the mathematical constant π with some bits flipped, i.e., it was generated from a completely deterministic process, and so, despite appearances, it was not random at all. This underlines the need for a formal theory that reconciles our heuristic notions of randomness with probabilistic principles.

The notion the Kolmogorov complexity is a relevant concept that arises from analyzing the randomness from an algorithmic standpoint. It focuses on the concept of *algorithmic randomness*, which is a measure that represents the length of the shortest computer program that can reproduce a given output, based on a Turing Machine tape. This approach to randomness is different from uncertainty. We now present some relevant definitions regarding Kolmogorov complexity.

Definition 1 *Let A be an algorithm. The algorithmic complexity of a sequence x in relation to A is denoted as $K_A(x)$ and it is defined as the minimum length of the programs (π) that can generate the sequence x using the algorithm A.*

Definition 2 *Let A be an algorithm with prefix-free domain of definition D (i.e., no word in D is the prefix of any other word in D). The prefix-free algorithmic complexity of a sequence x, denoted as $\tilde{K}_A(x)$, is the length of the shortest word π in D such that $A(\pi)$ equals x. If such π doesn't exist, then $\tilde{K}_A(x)$ is positive infinity.*

Proposition 1 *There exists an optimal prefix-free algorithm A_0 such that, for any other prefix-free algorithm A, there is a constant $C > 0$, where $\tilde{K}_{A_0}(x) \leq \tilde{K}_A(x) + C$.*

The prefix-free complexity $\tilde{K}(x)$ of the word x is defined as equal to the complexity $K_{A_0}(x)$ with respect to one fixed (for all considerations) optimal prefix-free algorithm A_0.

Definition 3 *A sequence x is deemed Kolmogorov-Chaitin random if it is incompressible, i.e., no initial segment of x can be compressed more than for a fixed finite number of bits. In other words, there exists a positive number b such that $\tilde{K}(x_{1:n}) \geq n - b$ for all n.*

Thus, this approach to randomness establishes that a sequence is random if there is no shorter computer program or algorithm that can produce it, rendering it incompressible. This aligns with our intuitive understanding of randomness as a lack of pattern or order that can be exploited to compress the data, as any causal relationship would render the sequence deterministic.

2.2 Information Theory

In this subsection, we provide a brief overview of the most relevant results from Information Theory that will be present in later sections. Our main topic of focus will be the notion of *entropy*, and how to quantify the amount of randomness is a given system.

Definition 4 *For a given random variable X with probability distribution p, for a given event $x \in X$ define the information content of x as*

$$h(x) = -log(p(x)) \tag{1}$$

The *information content*, as the name suggests, is a reasonable way of estimating the amount of information contained in a given event. A important point is the sign, which is used to make the function image from 0 to 1 positive, as the domain is a probability.

Following from this, we can naturally define the notion of Shannon entropy.

Definition 5 *For a given random variable X, with probability distribution p, define the Shannon entropy of X as*

$$H(X) = \mathop{\mathbb{E}}_{x \in X}[h(x)] = \sum_{x \in X} -p(x)log(p(x)) \tag{2}$$

The Shannon entropy is a natural way of quantifying the amount of randomness – or uncertainty – in a given random variable. This definition of entropy is just the expected value of the information contents of all events, and thus it can be seen as a way of accounting for the average information present in the set of all events. There are, however, other ways of defining entropy. We can define the *min-entropy* of a classical random variable as follows

Definition 6 *Let X be a classical random variable. The minimum-entropy (min-entropy) of X is defined as*

$$H_{min}(X) \coloneqq h(\max_{x \in X} p(x)) \tag{3}$$

This means that the min-entropy of X is defined as the information content of the most likely event in X, which in turn guarantees that such quantity is always upper-bounded by the Shannon Entropy of X. The min-entropy is often referred to as a conservative measure in the sense that it is trying to describe the randomness of a system solely based on the most likely outcome, and this is of particular interest for measuring the unpredictability of highly sensitive systems w.r.t. privacy, e.g., passwords.

The above definitions, although brief, will prove to be essential for the next sections, and for the main results discussed in this article.

2.3 Classical RNG and CSPRNG

Based on the definitions of randomness, we know that deterministic procedures cannot generate random values, but that doesn't exclude the possibility of creating pseudorandom number generators that are good enough for a given application. As a matter of fact, most of the current computer systems that rely on randomness actually depend on pseudorandom numbers. These classical RNG systems are, by definition, deterministic, since they generate the same output given the same initial seed. They are also periodic, since sequences of numbers will eventually repeat, and the length of distinct outputs before repetition is often referred to as *period*. Moreover, these systems are designed to be extremely efficient.

We now provide an example of one such classical RNG algorithm.

Definition 7 *The Linear Congruential Generator generates numbers via the following recurrence relation:*

$$X_{n+1} = (aX_n + c) \mod m \tag{4}$$

where X_n denotes the n-th element of the sequence, and a, c, m are constants.

Since this algorithm generates numbers using the modulus operation, the sequence of numbers will eventually repeat after a large enough number of iterations. If an adversary knows the algorithm architecture and has some samples or can discover/guess the seed, they can predict all future numbers in the sequence, breaking the security of any protocol that relies on such algorithm.

A clever approach for designing better RNGs is to counteract its two most critical flaws: periodicity and determinism. By seeding the algorithm with entropy collected from some physical source – say a radioactive decay, for instance – the seed itself will be generated from a certifiably random process, and thus any adversary will have little hope of guessing the original seed.

Most programmers in day to day tasks use algorithms like `Math.random()` or their preferred language equivalent for random number generation purposes, being careful in hiding/-choosing a good `seed`, such that attackers won't predict the next number, and periodically reseeding. However, it can be shown that one can trivially discover the algorithm's pattern and exploit it for malicious ends if no other ciphering and automatic reseeding is employed.

An example of this is the *javascript* random method, implemented by Google in the V8 Engine, that uses the *Xorshift algorithm* [Mar03], which, given two states, performs shift operations with special constants, in order to achieve pseudo randomness. However, since this algorithm consists of a known set of operations and unknown values, one can use a satisfability modulo theory solver – such as the *Z3 SMT solver* [Mic23] – to model possible seed values as theorems, and then check if their are satisfiable in order to compute possible solutions. Hence, one can easily break the security of such RNG system.

A way to address this vulnerability is to employ a Cryptographically Secure Pseudo-Random Number Generator (CSPRNG), which must satisfy two criteria:

- It must pass the "next-bit test", which means that if you know the first 'k' bits in a sequence, there shouldn't be any practical method (algorithm) that can guess the next bit (the (k+1)th bit) with a success rate that is meaningfully higher than pure chance (50%). This concept was demonstrated by Andrew Yao in 1982, who showed that if a

generator passes this test, it would also pass any other statistical tests for randomness that can be executed in a practical amount of time.

- It must be resistant to "state compromise extensions". This means that even if a part or all of its internal state (the information it uses to generate the next random number) is exposed or correctly guessed, it should still be impossible to rebuild the previously generated random numbers. Furthermore, if the generator accepts an entropy input (a source of randomness) during operation, knowing the state of this input shouldn't allow predicting future states of the CSPRNG.

For instance, if a CSPRNG creates output by computing the bits of the number π in a sequential order, starting from an unknown point in the binary representation of π, it might pass the next-bit test. That's because π seems to be a random sequence. However, this method also isn't cryptographically secure: if an attacker identifies the current state of the generator (the current bit of π being used), they could calculate all the preceding bits.

A good algorithm, employed by some Linux distributions, is the *Fortuna CSRNG* [Wik], employing a strategy of entropy accumulation and clever generation. However, if you know parts of the algorithm state and architecture, it may be possible to reverse the seed, but employing reseeding avoids the issue, as the attacker would need a SMT solver capable and fast enough to deal with the changes.

2.4 Quantum Information

In this subsection, our main objective is to formalize the many notions of entropy defined for quantum systems, as well as stating a proving some of the more useful theorems related to these quantities.

First, we define a quantum analogue for the Shannon entropy.

Definition 8 *Let ρ_A denote a density matrix on a quantum system A. The Von Neumann entropy of ρ_A is given by:*

$$H_\rho(A) := -tr(\rho \log \rho) = -\sum_{\lambda \in spec(\rho)} \lambda \log(\lambda) \tag{5}$$

where $\log \rho$ denotes the matrix logarithm of ρ, and $spec(\rho)$ denotes the spectrum of ρ. For a bipartite state ρ_{AB}, its Von Neumann entropy is given by:

$$H_\rho(A|B) = H_\rho(AB) - H_\rho(B) \tag{6}$$

Such definition of entropy merely translates the notion of Shannon entropy to a quantum system, i.e., measuring the average unpredictability of the outcomes of the system. We now define the notion of relative entropy.

Definition 9 *Let ρ_A, σ_A denote two density matrices on a quantum system A. The relative entropy of ρ_A to σ_A is given by:*

$$D(\rho_A||\sigma_A) := tr(\rho_A \log \rho_A) - tr(\rho_A \log \sigma_A) = tr(\rho_A(\log \rho_A - \log \sigma_A)) \tag{7}$$

The relative entropy can be seen as a quantum generalization of the Kullback-Leibler (KL) divergence of the states, and indeed, if $\rho_A \sigma_A = \sigma_A \rho_A$, i.e., they commute, then they will

be simultaneously diagonalizable by a set of orthonormal eigenvectos, with possibly distinct eigenvalues. We then get that

$$\begin{aligned} &\log \rho_A = V \mathrm{diag}(\{\log \lambda | \lambda \in \mathrm{spec}(\rho)\}) V^*, \log \sigma_A = V \mathrm{diag}(\{\log \theta | \theta \in \mathrm{spec}(\sigma)\}) V^* \\ &\quad \Rightarrow \log \rho_A - \log \sigma_A = V(\mathrm{diag}(\{\log(\lambda/\theta)\}) V^* \\ &\quad \Rightarrow \mathrm{tr}(\rho_A(\log \rho_A - \log \sigma_A)) = \sum_{i=1}^{n} \lambda_i \log(\lambda_i/\theta_i) \end{aligned} \tag{8}$$

where n is the dimension of the Hilbert-Space, log is taken base 2, and spec denotes the spectrum of an operator. This means that, if the density matrices commute, their relative entropy is just the standard KL divergence w.r.t. their eigenvalues. Moreover, we also define $D(\rho_A \| \sigma_A) = \infty$ when $\mathrm{supp}(\rho_A) \cap \ker(\sigma_A) \neq \{0\}$, where $\mathrm{supp}(\rho_A) = \{v \in \mathbb{C}^n | \rho_A v \neq 0\}$.

From the previous definition, we can naturally define the notion of maximum relative entropy.

Definition 10 *Let ρ_A, σ_A denote two density matrices on a quantum system A. The maximum relative entropy of ρ_A to σ_A is given by:*

$$D_{max}(\rho_A \| \sigma_A) := \inf\{\lambda \in \mathbb{R} : \rho_A \leq 2^{\lambda} \sigma_A\} \tag{9}$$

Finally, we can now define the notion of conditional min-entropy.

Definition 11 *Let ρ_{AB} denote a density matrix on a bipartite quantum system AB. The conditional min-entropy of A conditioned by B is defined as:*

$$H_{min}(A|B)_\rho := \sup_{\sigma_B} \{-D_{max}(\rho_{AB} \| I_A \otimes \sigma_B)\} \tag{10}$$

Intuitively, the conditional min-entropy describes the amount of extractable randomness system A has given the information available on B. In the case that A, B are classical systems, they are considered to be random variables, and both ρ_{AB} and σ_B are considered to be diagonal matrices on the same basis defining a probability distribution over the systems.

One can also deduce an interesting and useful link between the classical notion of min-entropy and the quantum one. Let Q be a quantum system with state ρ_x, where x depends on a classical random variable X, and let $p(x)$ denote the probability of x. We can consider a state from the system XQ as follows

$$\rho_{XQ} = \sum_x p(x) |x\rangle\langle x| \otimes \rho_x \tag{11}$$

where $\{|x\rangle|\}$ forms an orthonormal basis. Let $p_g(X|Q)$ be the probability that an agent guesses X when using an optimal measurement strategy. It can be shown that, if ρ_{XB} is a pure state, then

$$H_{\min}(X|B) = H_{\min}(X) \tag{12}$$

which, in other words, implies that

$$2^{-H_{\min}(X|B)} = \max_{x \in X} \{p(x)\} \tag{13}$$

As we will see in further sections, this previous equation will be extremely useful for proving certified randomness in the protocol described in [AH23].

As a final topic, we briefly discuss the concept of what is called the *Haar* measure on the group $U(N)$, for a given natural N. It can be shown that such group admits a unique probability measure that is translation-invariant, i.e., the *Haar* measure is invariant w.r.t. the group action of $U(N)$ – that being matrix multiplication –. Moreover, this is a manifestation of a more general phenomenon: any locally compact group has a *Haar* measure that is invariant under the group action, and since $U(N)$ is a topological group that is locally compact and separable, it satisfies such condition.

From the above definition, however, it is not obvious to see how would one constructively obtain a Haar random unitary matrix. One extremely interesting way is as follows:

1. First, start with an $N \times N$ matrix A with i.i.d. standard complex Gaussian random variables, i.e., fill each entry of the matrix with a value sampled independently from a single complex Gaussian distribution.

2. Compute the QR decomposition of A, where $A = QR$, Q being an unitary matrix, and R being an upper-triangular matrix. This can be done efficiently by using the Gram-Schmidt process on the columns of A.

3. The matrix Q will then satisfy $Q \sim \text{Haar}(N)$.

The above procedure is relatively simple, and can be efficiently implemented on a classical computer. We now state a Lemma that will be extremely important for deriving future results.

Lemma 1 *Let $C \in U(N)$ be an unitary matrix representing a quantum circuit, and let p_C denote the probability distribution over the N-dimensional Hilbert-space, defined by $p_C(s) = |\langle s|C|0\rangle^{\otimes n}|^2$. If $C \sim Haar(N)$, then*

$$\mathop{\mathbb{E}}_{C} \mathop{\mathbb{E}}_{s\sim p_C} [p_C(s)] = \frac{2}{N+1} \tag{14}$$

■

A full proof of the previous Lemma can be found in Section 3.4.2 from [AH23], but we've decided to omit it since it relies on a number of non-trivial results regarding the Dirichlet Distribution that we believe don't fit the scope of this article.

2.5 Quantum Complexity Theory

First, we begin by describing one of the most famous interactive proof systems, know as the Arthur-Merlin protocol. In such protocol, Arthur is the verifier and Merlin is the prover, i.e., Arthur has access to polynomial-time operations and wishes to determine if a given string belongs to a language by exchanging messages with Merlin, which in turn has unlimited power, however cannot be fully trusted. Moreover, Arthur is assumed to be running a probabilistic algorithm, i.e., at every step of the message-exchange with Merlin, his decisions are based on a given probability distribution. This means that Arthur can be thought of as possessing a RNG device.

We now define two natural complexity classes that arise from this protocol.

Definition 12 *Define MA (Merlin-Arthur) as the set of languages L for which there exists a polynomial-time deterministic algorithm V_L, and a polynomial p such that for every string $x \in \{0,1\}^n$:*

- $x \in L \Rightarrow \exists c \in \{0,1\}^{p(n)} : \mathbb{P}[V_L(x,c) accepts)] \geq 2/3$
- $x \notin L \Rightarrow \forall c \in \{0,1\}^{p(n)} : \mathbb{P}[V_L(x,c) accepts)] \leq 1/3$

Intuitively, the MA class contains all languages such that if a string is in the language, then there exists a polynomial-sized proof that Merlin can send Arthur to convince him to accept such string with high probability, otherwise there is no such proof that can convince Arthur to accept the string with high probability. Therefore, if a problem is in MA, then there is no good way of tricking Arthur into making a mistake, but there is always a way of convincing him to accept a right answer. We now define a similar complexity class that arises from the possibility of two messages being exchanged.

Definition 13 *Define AM (Arthur-Merlin) as the set of languages L for which there exists a polynomial-time deterministic algorithm V_L and polynomials p, q such that for every string $x \in \{0,1\}^n$:*

- $x \in L \Rightarrow \mathbb{P}_{s \in \{0,1\}^{q(n)}}[\exists c \in \{0,1\}^{p(n)} : V_L(x,c,s) accepts] \geq 2/3$
- $x \notin L \Rightarrow \mathbb{P}_{s \in \{0,1\}^{q(n)}}[\forall c \in \{0,1\}^{p(n)} : V_L(x,c,s) accepts] \leq 1/3$

In the case of AM, we are considering a two message Arthur-Merlin protocol: Arthur inputs x to his RNG and obtains a string y with polynomial-size w.r.t. x, and then sends y to Merlin, which in turn produces a proof c and then sends it back to Arthur, and finally Arthur decides whether or not to accept x via the polynomial-time deterministic algorithm V_L. Thus, if a problem is in AM, then there is a high-probability that, given an output of Arthur's RNG, there is a proof that Merlin can produce that convinces Arthur of accepting x, and there is a low-probability that Merlin can produce any proof that convinces Arthur of making a mistake.

We now define the quantum analogues of the classes MA and AM.

Definition 14 *Define QMA (Quantum Merlin-Arthur) as the set of all languages L for which there exists a quantum polynomial-time algorithm V_L and a polynomial p such that for every string $x \in \{0,1\}^n$:*

- $x \in L \Rightarrow \exists \rho$ *quantum state on* $p(n)$ *qubits such that* $\mathbb{P}[V_L(x,\rho) accepts] \geq 2/3$
- $x \in L \Rightarrow \forall \rho$ *quantum state on* $p(n)$ *qubits such that* $\mathbb{P}[V_L(x,\rho) accepts] \leq 1/3$

In the case of QMA, Merlin can send a quantum state ρ to Arthur instead of a classical string of bits.

Definition 15 *Define QCAM (Quantum-Classical Arthur-Merlin) as the set of languages L for which there exists a quantum polynomial-time algorithm V_L and polynomials p, q such that for every string $x \in \{0,1\}^n$:*

- $x \in L \Rightarrow \mathbb{P}_{s \in \{0,1\}^{q(n)}}[\exists c \in \{0,1\}^{p(n)} : V_L(x,c,s) accepts] \geq 2/3$
- $x \notin L \Rightarrow \mathbb{P}_{s \in \{0,1\}^{q(n)}}[\forall c \in \{0,1\}^{p(n)} : V_L(x,c,s) accepts] \leq 1/3$

Note that the only difference between QCAM and AM is the nature of the algorithm V_L, i.e., it is considered to be a quantum algorithm, whereas the difference between QMA and AM is both due to the nature of V_L and due to the fact that Merlin can now send a quantum state ρ

to Arthur instead of a classical string. We also denote $\mathsf{QMA}(T)$ as the class QMA with query complexity T, i.e., Arthur's algorithm V_L is $O(T)$ (the same notation is used for the other complexity classes). Finally, we define $\mathsf{QCAMTIME}(T)$ as the generalization of QCAM where Arthur's algorithm V_L is also $O(T)$ but the communication is still required to be polynomial, and $\mathsf{QCAMTIME}(T)/\mathsf{q}(A)$ is the same but Arthur receives A bits of quantum advice that only depend on the size of the input string.

3 Quantum Random Number Generators

In this section, we formalize the concept of Quantum Random Number Generators, and then provide an interesting case study regarding quantum devices. The main references for subsection 3.1 were [MYC+16, JJNJ21], and for 3.2 were [**?**].

3.1 QRNG

A Quantum Random Number Generator (QRNG) is any RNG system using quantum systems to assist in the random number generation process. The fundamental principle behind hardware QRNGs is the assumption that there is no deterministic way of determining to which state a given quantum state will collapse. This implies that the measurement of a superposition state, such as a superposition of qubits, results in the production of a fundamentally random output. Two key stages need to be distinguished in this process:

1. Setup: The preparation of the initial state, which requires trust from the user that the system is indeed in the superposition state
2. Generation: The process of measuring a series of states, generating the desired random numbers.

In order to build a secure QRNG, one must guarantee the quantum state is indeed the expected initial state, and also guarantee that the prepared system was not affected by external agents, which can collapse the initial state and potentially compromise the security of the whole system. Concerning these questions, QRNGs can be broadly divided into three categories based on the hardware implementation, offering a trade off between speed and reliability.

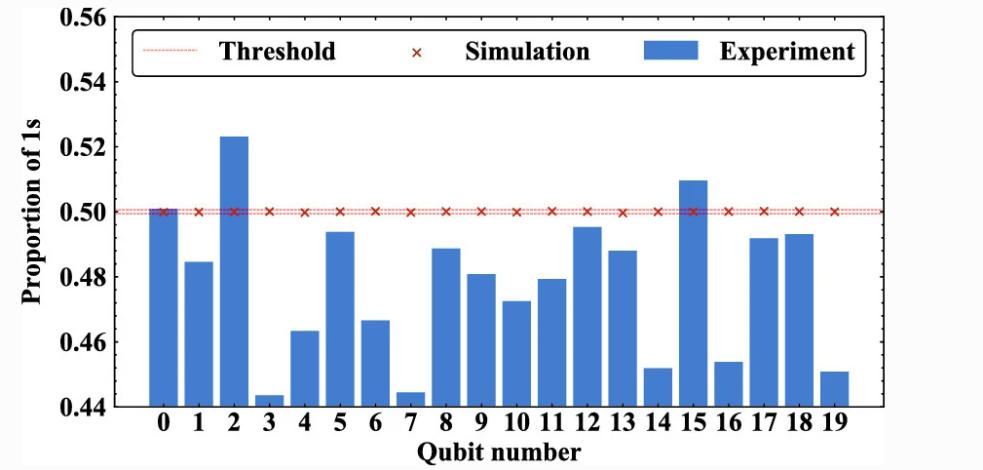
1. **Practical Quantum Random Number Generators:** These are fully trusted, calibrated devices where randomness hinges on the accurate modeling and execution of the physical quantum process. While these devices typically generate numbers at a moderate speed and are relatively inexpensive, they often mix quantum randomness with classical noise, necessitating the correct modeling of the quantum process for noise removal. A relevant flaw is the need to trust in the device and its components is integral for the security of these generators.
2. **Self-Testing Quantum Random Number Generators:** These generators perform tests on the generated sequence due to limited confidence in the implemented physical process. The testing could rely on classical tests or on the verification of quantum entanglement, via Bell inequalities. Such generators are also known as device-independent QRNGs. These generators are often slower or require additional complex testing apparatus due to the complexity of the testing process.

3. **Semi-Testing Quantum Random Number Generators:** These generators are a compromise between the previous two categories. In these devices, some level of confidence in the implementation reduces the need for extensive testing, optimizing speed parameters at the cost of confidence in the generated randomness. While some components in these devices are considered safe due to precise characterization, others need more extensive testing.

However, a key component is still missing: how to know if the numbers were generated by an honest quantum device? This question is the bedrock of certifiable quantum randomness, and we heavily discuss it in further sections.

3.2 Evaluating a PQRNG on a real quantum computer

The article [TS21] provides a comprehensive analysis and case study of using a real quantum computer to generate large sequences of random numbers, and then submitting them to statistical tests. 579 samples were obtained from the IBM 20Q Poughkeepsie, featuring 20 qubits, each producing 579 samples with a length of 8192 bits. The next figure illustrate the author's results.



The authors conclusion was that the random number generation procedure failed to surpass the testing threshold, as seen in the figure. Both real device and simulator samples underwent a statistical review to assess bias and patterns. The min-entropy was calculated to examine the uniformity of each sample. The changes in min-entropy indicated distinct characteristics of the qubits. A notable decrease in min-entropy was observed in qubit [17]. A histogram depicting the ratio of 1s in the sequences of 4,743,168-bits, generated by each qubit, suggested that none of the qubits yielded an adequate proportion of 1s. However, the stability of each qubit was evaluated using time-series data from the proportion of 1s, and qubits [0] and [12] were found to be relatively steady. Eventually, the 529 samples from the 20 qubits underwent eight tests from NIST SP 800-22. No qubit met the suite test standards, but the results indicated that qubits [0] and [12] came closest to the ideal proportion of passed samples in each test.

This example illustrates that modern quantum computers are far away from the perfect devices discussed in theory, and thus can often produce outputs that may seem to contradict the expected results predicted by quantum computing.

4 Certified Randomness from Quantum Supremacy

This section consists of an in-depth discussion regarding the recent findings of Scott Aaronson and Shih-Han Hung in [AH23]. The authors propose a way of reliably generating cryptographically certified random bits, i.e., a QRNG system that allows the users to verify its legitimacy. This is an extremely relevant property for enhancing user's trust in security and cryptographical systems in general, since the existence of a backdoor is a direct threat to user's privacy.

The aforementioned article is, however, quite long and detailed, with more than 80 pages of content, and so, instead of trying to summarize the entirety of the article, we've decided to focus on the sections that we consider to be the most relevant and most descriptive of their overall findings, namely sections 3 and 5 of [AH23]. Section 3 provides useful mathematical facts, that are then used in Section 5 to describe a weaker version of the paper's main result under the assumption that the quantum device does not share an entangled state with a possible adversary trying to predict the device's output. Overall, the findings of these sections together with the mathematical tools and concepts used to prove them are a perfect fit for the scope of our work. The main sources for all of the following subsections were [AH23, AAB+19, NRK+18].

Over the next subsections, we let $N = 2^n$ denote the dimension of the Hilbert-space, and let $\mathcal{D}$ be a probability distribution over $U(N)$, where $U(N)$ denotes the group of $N \times N$ unitary matrices. Since every quantum circuit can be represented by a unitary matrix, defining a distribution over $U(N)$ equates to defining a distribution over the set of all possible quantum circuits. We also use $x \sim P$ to denote that x is sampled from the distribution P.

4.1 Quantum Supremacy and the LXEB Problem

We now turn our attention to the main problem studied in [AH23, AAB+19], know as *Linear Cross-Entropy Benchmarking* (LXEB).

Definition 16 (Linear Cross-Entropy Benchmarking) *Let $\mathcal{D}$ be a probability distribution over $U(N)$, C drawn from $\mathcal{D}$ and $p_C(s) = |\langle s|C|0\rangle^{\otimes n}|^2$. For a set of samples $s_1, s_2, ..., s_k$, $s_i \in \{0,1\}^n$, define $F_{LXEB(b,k)}$ as:*

$$F_{LXEB(b,k)}(s_1, ..., s_k) := \frac{N}{k}\left(\sum_{i=1}^{k} p_C(s_i)\right) - b \tag{15}$$

The LXEB(b,k) problem consists of generating samples such that $F_{LXEB(b,k)} \geq 0$, i.e., ensuring that

$$\frac{1}{k}\sum_{i=1}^{k} p_C(z_i) \geq \frac{b}{N} \tag{16}$$

The LXEB was defined in the context of the experiments of *Quantum Supremacy*, performed by Google's Sycamore, and most recently by USTC's Jiuzhang. The main idea of the experiment, which is often referred to as Random Circuit Sampling (RCS), can be summarized as follows: a user generates a random quantum circuit C and then sends it to a server running a quantum computer (QC), the server then applies the circuit to the $|0\rangle^{\otimes n}$ state, measures it

with the computational basis, and then returns the output string to the user – and the server response needs to be fast –. This process is repeated multiple times, and then the user applies a statistical test to check if the results are indeed consistent with a QC running the circuit C. The LXEB is one such statistical test, where the user receives the output strings and then computes $F_{\text{LXEB}(b,k)}$ in the classical computer, and if the test succeeds, then the user can be certain that, with high probability, the server's QC did indeed run C.

Currently, there is no known algorithm for efficiently simulating a QC on a classical computer, and so in the context of the RCS task, for a large enough number of qubits, a classical strategy has little to do other than to randomly sample strings from the uniform distribution over $\{0,1\}^n$, and then send them to the user as output. This strategy clearly works in the case where $b = 1$, since the expected value of the sum of the probabilities would be exactly k/N, however, if $b > 1$, one would expect any efficient classical strategy to fail the LXEB problem. On the other hand, from Lemma 1, we see that a noiseless QC would have an expected LXEB fidelity of $F_{\text{LXEB}(b,k)} = 2(N/(N+1)) - b$, and so for values of $b < 2$, a noiseless QC is expected to solve LXEB(b,k) with high probability, given that the number of samples is large enough. For this reason, solving the RCS task via the LXEB problem can be seen as a way of achieving "*Quantum Supremacy*": if we set $b > 1$, for sufficiently large number of samples k, only a QC would be able to return samples that pass the LXEB(b,k) test, i.e., there is a real advantage of using a QC for this specific task. Google's 2019 experiment [AAB+19] achieved such result using $b \approx 1.002$, and 54 qubits in their Sycamore processor.

One of the most important properties of the LXEB is that the probabilities $p_C(s_i)$ can be computed directly with $|\langle s|C|0\rangle^{\otimes n}|^2$. This is extremely useful since it means that one does not need to make multiple runs of the circuit in order to create a large enough sample for approximating the true probability with the empirical probability. As a direct consequence, if one has access to the samples z_i and the unitary matrix representing C, the probabilities can be calculated on a classical computer. This plays a key-role in the cryptographically-certified QRNG envisioned by Aaronson and Hung: if we imagine a QC running the circuit C on a server and outputting the samples z_i to a user, if they have access to the unitary matrix representing C and the parameter b, than they can compute $F_{\text{LXEB}(b,k)}$. Moreover, as we will see in further sections, the authors of [AH23] show that, under some complexity assumptions, one can derive a lower bound to the number of random bits in the output of a quantum algorithm that solves the LXEB problem. *This implies that the user can compute* $F_{\text{LXEB}(b,k)}$ *and use the result to be sure that some of the received bits are indeed random.* Aaronson and Hung's protocol can then be summarized as follows:

1. A user generates a sequence of n-qubit circuits $C_1, C_2, ..., C_M$ pseudorandomly w.r.t. a small random seed and sends them to a server.

2. For every C_i, the server quickly returns k n-bit strings. If the server is indeed honest, then the strings are samples from the output distribution of $C_i|0\rangle^{\otimes n}$.

3. The user uses their seed to choose a subset of the circuits, and then checks if the k samples returned from the QC pass the LXEB$(b,k), b \in (1,2)$.

4. If the checks pass, then the user feeds the strings into a classical randomness extractor to get nearly n pure random bits.

However, it is important to stress the following: the unitary matrix representing C is a $2^n \times 2^n$ matrix, and so computing $F_{\text{LXEB}(b,k)}$ would still take exponential time on the number of qubits. This is one of the downsides of the protocol, since it means that for a relatively small number of qubits, e.g., 50 qubits, the user would require a supercomputer in order to compute $F_{\text{LXEB}(b,k)}$. On the other hand, the author's protocol is also one of the few practical applications of quantum computing that inherently requires a QC and is also physically realizable with today's QCs.

The LXEB has many relevant practical properties when it comes to the implementation of a QC, however such properties are not within the scope of this article, and so we refer the reader to [AAB+19, NRK+18] for further information. It also has possible limitations and flaws that are extremely relevant for designing future experiments, however again these are beyond the scope of this article, and so we also refer the reader to [GKC+21, AGL+23] for further insight.

4.2 The Hardness Assumption

We begin by defining an essential problem to the results in [AH23], known as Long List Quantum Supremacy Verification (LLQSV). In the following descriptions, $\mathcal{D}$ will denote a probability distribution over $U(N)$, and such term will often be omitted when it is either obvious from the context or irrelevant to the discussion.

Definition 17 (Long List Quantum Supremacy Verification) *Let $M = O(N^3)$. Suppose we are given access to n-qubit quantum circuits $C_1, ..., C_M$ drawn independently from $\mathcal{D}$, and strings $s_1, ..., s_M, s_i \in \{0,1\}^n$, and let p_{C_i} denote the output distribution of C_i, $p_{C_i}(z) = |\langle z|C_i|0^n\rangle|^2$. The LLQSV($\mathcal{D}$) problem consists distinguishing between the following cases:*

- ***Yes-case**: each s_i is sampled from C_i, i.e., $s_i \sim p_{C_i}$*
- ***No-case**: each s_i is sampled from a uniform distribution over $\{0,1\}^n$ (hence independent of C_i)*

The LLQSV problem then consists of building an algorithm that, given a long list of circuits (exponential on the number of qubits n), can distinguish between strings that came from the output distribution of such circuits and strings sampled uniformly from the set of all n-bit strings. Note that this problem is closely related to the aforementioned *Quantum Supremacy* experiments: we can think of the input circuits $C_1, ..., C_M$ as the circuits sent from the user to the server, and the strings $s_1, ..., s_M$ as the server's response. The problem then consists of designing a quantum algorithm for the server that can efficiently convince the user to accept the response, and in the context of the LXEB test, this can be thought as ensuring that the output strings pass the LXEB. In the case of an honest server, just running the circuits on a QC and returning the measured outputs would suffice, however we do need to account for a dishonest server.

In a classical scenario, one would expect that there is no way for a classical algorithm to distinguish between a *Yes-case* and a *No-case* in polynomial time, since in order to determine if a given s_i was sampled from C_i, one would need to simulate C_i, and that would classically require exponential time on the number of qubits. This of course depends on the assumption that there is no efficient classical algorithm for simulating a QC, which has not yet been proven, however it is mostly a consensus amongst experts.

In the quantum scenario, the question about if there exists a quantum algorithm that does not sample from the circuit's output distribution but still convinces the user to accept the strings remains open. In order to derive their results, however, Aaronson and Hung *assume that there is no such algorithm*. They refer to this assumption as the Long List Hardness Assumption ($\text{LLHA}_B(\mathcal{D})$) w.r.t. a parameter $B < n$ and a probability distribution $\mathcal{D}$ over $U(N)$.

Assumption 1 (Long List Hardness Assumption) *Let $\mathcal{D}$ be a probability distribution over $U(N)$, then $LLQSV(\mathcal{D}) \notin \mathit{QCAMTIME}(2^B n^{O(1)})/q(2^B n^{O(1)})$.*

The LLHA is assuming that there is no Arthur-Merlin protocol where Arthur sends Merlin an $n^{O(1)}$-bit classical string, then Merlin replies with an $n^{O(1)}$-bit proof that convinces Arthur of the *Yes-case*, even if Arthur can do the verification using a 2^B-time quantum algorithm, with 2^B qubits of quantum advice. In other words, the authors are assuming that there is no quantum algorithm that can efficiently trick Arthur into making a mistake, and so if a quantum algorithm is efficiently solving the LLQSV problem, it must be doing so by sampling. As we will see in future sections, this will allow us to derive a lower-bound for the number of random bits in the output of any quantum device that passes the LXEB problem.

4.3 An overview of the proof

Here we provide an outline of the strategy used by Aaronson and Hung to prove the main theorem of Section 5 of [AH23], that result being: *assuming the LLHA, then any QC solving the LXEB problem must output at least $\Omega(n)$ random bits*, i.e., the protocol summarized in section 4.1 does indeed work under the LLHA. *The proof will follow by contradiction*, and will be organized as follows:

(1) In subsection 4.4 we define a classical Arthur-Merlin protocol that has extremely useful properties for our purposes, and then in subsection 4.5 we explain how to adapt such protocol to the quantum scenario. More importantly, we show how, given a quantum Merlin-Arthur protocol satisfying some technical requirements, one can create a quantum Arthur-Merlin protocol with bounded gap.

(2) Then, in subsection 4.6, we assume the existence of a quantum algorithm that solves the LXEB with low-entropy, i.e., an algorithm whose outputs have less than $\Omega(n)$ random bits. This can be seen as assuming that there is a way of designing a dishonest QC server, in the context of the RCS task described in subsection 4.1, that can pass the LXEB test made by the user. We then proceed to derive a series of useful results that will allow us to prove that our quantum algorithm gives birth to a Merlin-Arthur protocol satisfying our desired technical requirements.

(3) After this, in subsection 4.7, we show how to create a quantum Arthur-Merlin algorithm solving the LLQSV problem – described in subsection 4.2 – in polynomial time.

(4) Finally, in subsection 4.8, we show that this directly contradicts the LLHA, and so indeed if the LLHA holds, then any quantum device solving the LXEB problem must output nearly n pure random bits.

4.4 The Goldwasser-Sipser protocol

In order to prove their main result, Aaronson and Hung heavily rely on the Goldwasser-Sipser protocol (GSP) for approximate counting [GS86]. In this subsection, we briefly describe such protocol and its useful properties, and as a main source, we will be using Professor Feigenbaum's lecture notes on computational complexity [Fei15].

The GSP protocol consists of a Arthur-Merling protocol, i.e., two messages are exchanged between Arthur and Merlin: Arthur first makes a sample and sends it to Merlin, which in turn must produce a proof and send it back to Arthur. Arthur has access to a set $S \subseteq \{0,1\}^n$ and an integer K. Merlin's goal is to send a proof that convinces Arthur that $|S| \geq K$. The protocol then will have the following property: if Merlin is telling the truth, then Arthur accepts it with high probability, and if Merlin is making a highly incorrect claim, i.e., if $|S| \leq K/2$, then Arthur rejects it with high probability. A pseudocode description of the GSP is given in Algorithm 1.

Algorithm 1 Goldwasser-Sipser protocol
Input: $S \subseteq \{0,1\}^n, K \in \mathbb{Z}$
Assume: let $\mathcal{H}_{n,R} = \{f : \{0,1\}^n \mapsto \{0,1\}^R\}$ be a family of pairwise-independent hash-functions, where $2^{R-2} < K \leq 2^{R-1}$

Arthur chooses $f \in \mathcal{H}_{n,R}$ and $y \in \{0,1\}^R$
Arthur sends **Merlin** the pair (f, y)
Merlin finds $x \in S : f(x) = y$
Merlin sends **Arthur** the pair (x, π), where π is a proof that $x \in S$
Arthur accepts if $f(x) = y$ and π is valid, otherwise rejects

It can be shown that such protocol indeed achieves the aforementioned property, although we will not prove this here directly. However, we will prove a similar result to a quantum analogue of the GSP in the following subsections.

4.5 Bounding the gap of the Quantum GSP

Let $\mathcal{D}$ denote a distribution over $U(N)$, and $\mathcal{U}$ denote the uniform distribution over the set $\{0,1\}^n$. Consider the same scenario as in definition 17: we have access to M tuples (C_i, s_i), with $C_i \sim \mathcal{D}$, and $s_i \sim p_{C_i}$ or $s_i \sim \mathcal{U}$. Let V be a Quantum Merlin-Arthur protocol with running time $2^B n^{O(1)}$ for the LLQSV problem, and assume that V accepts at least K tuples from the *Yes-case* of LLQSV – which gives a lower bound on the number of correct guesses from the algorithm –, and accepts at most $(1 - \Omega(\epsilon))K$ tuples from the *No-case* of LLQSV – which in turn gives an upper bound on the number of incorrect guesses of the algorithm –, both with probability $1 - 2^{-\Omega(n)}$. From this, we can create a quantum analogue of the GSP using V, described in Algorithm 2, and referred to as Quantum Goldwasser-Sipser protocol (QGSP).

We are now interested in finding a bound for what is known as the *protocol gap*. This gap is defined as the difference between the probabilities of the protocol correctly and incorrectly accepting an arbitrary instance. In our case, we will prove that the gap of the QGSP is at least $\epsilon^2/4\alpha$, for parameters $\epsilon > 0, \alpha > 1$.

Lemma 2 *Given real numbers $\alpha > 1, \epsilon \in (0,1]$, and $K \in \mathbb{Z}$, there exists a Quantum Arthur-Merlin protocol which, given a set S as input, determines if $K \leq |S| \leq \alpha K$ or $|S| \leq (1-\epsilon)K$*

Algorithm 2 Quantum Goldwasser-Sipser protocol
Input: $(C_1, s_1), ..., (C_M, s_M)$, real parameters $\alpha > 1, R$ and $K \in \mathbb{Z}$

Arthur chooses a random hash function $f : [M] \mapsto [R]$ and $y \in [R]$
Arthur sends **Merlin** the pair (f, y)
Merlin finds $x \in [M] : f(x) = y$
Merlin sends **Arthur** the pair (x, π), where π is a proof that $x \in [M]$
Arthur accepts if V accepts (C_i, s_i) with proof π and if $f(x) = y$, otherwise rejects

with gap at least $\epsilon^2/4\alpha$, using a hash function of range size $R = 2\alpha K/\epsilon$.
Proof. As seen in the definitions of section 2.5, in the context of an Arthur-Merlin protocol, the probability of accepting a given input is determined by

$$\mathop{\mathbb{P}}_{f,y}[\exists x \in S, f(x) = y] \tag{17}$$

for a random hash function f of range size R, since it suffices for Merlin to find a string $x \in S$ such that $f(x) = y$. Thus, we can compute the gap by obtaining both a lower and upper bound to this probability.

From the union bound inequality, we know that given events $E_1, ..., E_n$, we have that

$$\mathbb{P}[\bigcup_i E_i] \leq \sum_i \mathbb{P}[E_i] \tag{18}$$

Note that the event $E = \exists x \in S$ is the same as $\bigcup_i E_i$, where E_i is the event that $x_i \in S$, for all $x_i \in \{0,1\}^n$. Thus, we have that

$$\begin{aligned} \mathop{\mathbb{P}}_{f,y}[\exists x \in S, f(x) = y] &= \mathop{\mathbb{P}}_{f,y}[\bigcup_{x_i \in \{0,1\}^n} [x_i \in S, f(x) = y]] \\ &\leq \sum_{x_i \in \{0,1\}^n} \mathop{\mathbb{P}}_{f,y}[x_i \in S, f(x) = y] \\ &= \sum_{x_i \in S} \mathop{\mathbb{P}}_{f,y}[f(x) = y] = \frac{|S|}{R} \end{aligned} \tag{19}$$

since $\mathbb{P}[x_i \in S, f(x) = y] = 0$ for $x_i \notin S$, and so we obtain an upper bound to Equation 17.

For the lower bound, we recall the inclusion-exclusion principle: given events $E_1, ..., E_n$, we have that

$$\mathbb{P}[\bigcup_i E_i] \geq \sum_i \mathbb{P}[E_i] - \sum_{i<j} \mathbb{P}[E_i \cap E_j] \tag{20}$$

We now apply such principle as follows

$$\begin{aligned} \mathop{\mathbb{P}}_{f,y}[\exists x \in S, f(x) = y] &\geq (\sum_{x_i \in S} \mathop{\mathbb{P}}_{f,y}[f(x) = y]) - (\sum_{x \leq z; x, z \in S} \mathop{\mathbb{P}}_{f,y}[f(x) = f(z) = y]) \\ &= \frac{|S|}{R} - \binom{|S|}{2} \frac{1}{R^2} = \frac{|S|}{R}(1 - \frac{|S| - 1}{2R}) \\ &> \frac{|S|}{R}(1 - \frac{|S|}{R}) \end{aligned} \tag{21}$$

therefore obtaining the desired lower bound.

Now we consider the case where $K \leq |S| \leq \alpha K$. By Equation 21, we obtain

$$\begin{aligned} C_1 := \mathop{\mathbb{P}}_{f,y}[\exists x \in S, f(x) = y] &> \frac{|S|}{R}(1 - \frac{|S|}{R}) \\ &\geq \frac{K}{R}(1 - \frac{\alpha K}{R}) \\ &\geq \frac{K}{R}(1 - \frac{\epsilon}{2}) \end{aligned} \tag{22}$$

since $R = 2\alpha K/\epsilon$. For the other case, i.e., $|S| \leq (1-\epsilon)K$, we obtain by Equation 19 that

$$C_2 := \mathop{\mathbb{P}}_{f,y}[\exists x \in S, f(x) = y] \leq \frac{|S|}{R} \leq \frac{K}{R}(1-\epsilon) \tag{23}$$

The gap is then given by

$$C_1 - C_2 > \frac{K}{R}(1 - \frac{\epsilon}{2}) - C_2 \geq \frac{K}{R}(1 - \frac{\epsilon}{2}) - \frac{K}{R}(1-\epsilon) = \frac{\epsilon^2}{4\alpha} \tag{24}$$

Therefore, the gap of the protocol is at least $\frac{\epsilon^2}{4\alpha}$, as desired. ∎

4.6 A low-entropy device solving the LXEB

In this subsection, we assume the existence of a low-entropy quantum device solving the LXEB problem. Our goal now is to show that this device satisfies the properties of the quantum algorithm V described in the last subsection, in order for use the QGSP for LLQSV problem, together with Lemma 2, to show that such algorithm violates the LLHA, arriving at a contradiction.

Assumption 2 (Low-entropy device solving the LXEB) *Let $\mathcal{A}$ be a quantum device solving LXEB(b,k) with probability q over choices of $C \sim \mathcal{D}$. By definition, $\mathcal{A}$ outputs k-tuples $(s_1, ..., s_k)$ that come from the output distribution of C. We further assume that, with probability p, the min-entropy of $(s_1, ..., s_k)$ w.r.t. C is at most $B/2$, i.e.,*

$$\mathbb{P}[-\log(\max_i p_C(s_i)) \leq \frac{B}{2}] = p \tag{25}$$

which can be rearranged into

$$\mathbb{P}[\max_i p_C(s_i) \geq \frac{1}{2^{B/2}}] = p \tag{26}$$

We also assume that that p, q satisfy

$$p > \frac{b}{b-1}(1-q) + \epsilon \tag{27}$$

Recall that B is a parameter such that the Quantum Merlin-Arthur algorithm V defined in the QGSP runs on $2^B n^{O(1)}$ time, which is also the same parameter defined for the LLHA$_B(\mathcal{D})$. Now, define the random variable $Y_\tau(C, s_i), \tau \in [0,1], s \in \{0,1\}^n$ as

$$Y_\tau(C,s) := \begin{cases} 1 & \text{if } \exists O = (z_1, ..., z_k) : \mathbb{P}[\mathcal{A}(C) = O] \geq \frac{\tau}{2^{B/2}} \text{ and } s \in (z_1, ..., z_k) \\ 0 & \text{otherwise.} \end{cases} \tag{28}$$

This random variable returns 1 if there is at least one k-tuple in the output distribution of $\mathcal{A}(C)$ that has probability at least $\tau/2^{B/2}$, and one of the elements of the k-tuple equals to s. In the next Lemma, we will also use $O \sim \mathcal{A}(C)$ to denote a k-tuple samples from the output distribution of $\mathcal{A}(C)$. We now define two expectations w.r.t. $Y_\tau(C,s)$ as follows

$$\begin{aligned} \mu_0(\tau) &:= \mathop{\mathbb{E}}_{C\sim\mathcal{D},s\sim\mathcal{U}}[Y_\tau(C,s)] \\ \mu_1(\tau) &:= \mathop{\mathbb{E}}_{C\sim\mathcal{D},s\sim p_C}[Y_\tau(C,s)] \end{aligned} \tag{29}$$

Finally, we define the following probability

$$p(\tau) := \mathop{\mathbb{P}}_{C\sim\mathcal{D}}[\max_d \mathbb{P}[\mathcal{A}(C)=d] \geq \frac{\tau}{2^{B/2}}] \tag{30}$$

This is the probability that $\mathcal{A}(C)$'s maximum probability is at least $\tau/2^{B/2}$, w.r.t. all $C \sim \mathcal{D}$. Moreover, if we model $\mathcal{A}(C)$ as a random variable, note that we can use Equation 26 to conclude the following

$$\begin{aligned} \max_d \mathbb{P}[\mathcal{A}(C)=d] \geq \frac{\tau}{2^{B/2}} &\iff -\log(\max_d \mathbb{P}[\mathcal{A}(C)=d]) \leq -\log(\frac{\tau}{2^{B/2}}) \\ &\iff H_{\min}(\mathcal{A}(C)) - \frac{B}{2} \leq -\log(\tau) \\ &\Rightarrow p(\tau) = \mathop{\mathbb{P}}_{C\sim\mathcal{D}}[H_{\min}(\mathcal{A}(C)) - \frac{B}{2} \leq -\log(\tau)] \end{aligned} \tag{31}$$

This means that the probability defined in Equation 30 is the probability that the gap between the min-entropy of the output samples $(s_1,...,s_k)$ and $B/2$ is at most $-\log(\tau)$. Also note that, since $-\log(1)=0$, we can combine Equations 31 and 25 to conclude that $p(1)=p$, i.e., if we set $\tau=1$, the probability $p(\tau)$ is measuring exactly the probability that our outputs have min-entropy at most $B/2$.

We can now also provide an intuitive interpretation for the expressions defined in Equations 28 and 29. Note that, by definition, if $Y_\tau(C,s)=1$ then there exists an output $O=(z_1,...,z_k)$ s.t. $\mathbb{P}[\mathcal{A}(C)=O] \geq \tau/2^{B/2}$, which in turn implies by Equation 31 that $H_{\min}(\mathcal{A}(C)) - B/2 \leq -\log(\tau)$, i.e., there is at least one output of our algorithm such that the gap between its min-entropy and $B/2$ is at most $-\log(\tau)$, and such that there is an output of $\mathcal{A}(C)$ that contains s, i.e., this implies that $p(\tau) > 0$. Therefore, if $Y_1(C,s)=1$ then the output distribution of $\mathcal{A}(C)$ indeed has min-entropy at most $B/2$ and $p(1)=1$.

The expectation $\mu_0(\tau)$ then can be seen as the expected number of output distributions $\mathcal{A}(C)$ that produce outputs with gap between their min-entropy and $B/2$ at most $-\log(\tau)$, and that can contain the string s, for all strings s in the uniform distribution, and for all circuits C over $\mathcal{D}$. The expectation $\mu_1(\tau)$ computes a similar quantity, but this time w.r.t. strings sampled from the output distribution of the circuit C. Also, we highlight that $\mu_0(1), \mu_1(1)$ compute the expected number of output distributions that produce outputs with min-entropy at most $B/2$, w.r.t. their respective string distributions.

The expectations $\mu_0(\tau), \mu_1(\tau)$ are closely related to the $\text{LLHA}_B(\mathcal{D})$ task, and we can interpret their ratio as the ratio between the expected number of output distributions that respect the assumption made in Equation 25 and the ones that don't. We now seek to bound the ratio $\mu_1(\tau)/\mu_0(\tau)$ using $p, q, p(\tau)$.

Lemma 3 *If $\tau \in [0,1]$ and $\mu_0(\tau), \mu_1(\tau), p(\tau)$ are defined as above, then*

$$\frac{\mu_1(\tau)}{\mu_0(\tau)} \geq b \cdot \frac{p(\tau) + q - 1}{p(\tau)} \tag{32}$$

Proof. Recall that, by definition, we have to analyze two cases w.r.t. the string s in the context of the LLQSV problem: the *yes case*, where the $s \sim p_C$, and the *no case*, where $s \sim \mathcal{U}$. Recall also that V is defined to be a quantum Merlin-Arthur algorithm for the LLQSV problem.

First, define

$$G_\tau(C) \coloneqq \chi[\max_d \mathbb{P}[\mathcal{A}(C) = d] \geq \frac{\tau}{2^{B/2}}] \tag{33}$$

that is, the indicator function for the event that the maximum probability of $\mathcal{A}(C)$ is at least $\tau/2^{B/2}$ for a fixed $C \sim \mathcal{D}$. Note that, by Equation 30

$$p(\tau) = \mathbb{P}_{C\sim\mathcal{D}}[G_\tau(C) = 1] = \mathbb{P}_{C\sim\mathcal{D}}[\exists d : \mathbb{P}[\mathcal{A}(C) = d] \geq \tau/2^{B/2}] \tag{34}$$

We can now analyze each case individually:

- For the *no case*, $s \sim \mathcal{U}$, hence

$$\begin{aligned} \mu_0(\tau) &= \mathop{\mathbb{E}}_{C\sim\mathcal{D}, s\sim\mathcal{U}}[Y_\tau(C,s)] \\ &= \mathop{\mathbb{P}}_{C\sim\mathcal{D}, O\sim\mathcal{A}(C), s\sim\mathcal{U}}[G_\tau(C) = 1, s \in O] \\ &= \mathop{\mathbb{P}}_{C\sim\mathcal{D}}[G_\tau(C) = 1]\frac{k}{N} = p(\tau) \cdot \frac{k}{N} \end{aligned} \tag{35}$$

- For the *yes case*, $s \sim p_C$. Let $\Theta(C,O)$ denote a verifier for an instance of the LXEB(b,k) problem, that is, let

$$\Theta(C,O) \coloneqq \begin{cases} 1 & \text{if } \sum_{i=1}^k p_C(z_i) \geq \frac{bk}{N}, \text{ where } O = (z_1, ..., z_k) \\ 0 & \text{otherwise.} \end{cases} \tag{36}$$

  Hence, we have that

$$\begin{aligned} \mu_1(\tau) &= \mathop{\mathbb{E}}_{C\sim\mathcal{D}, s\sim p_C}[Y_\tau(C,s)] \\ &= \mathop{\mathbb{P}}_{C\sim\mathcal{D}, O\sim\mathcal{A}(C), s\sim p_C}[G_\tau(C) = 1, s \in O] \\ &\geq \mathop{\mathbb{P}}_{C\sim\mathcal{D}, O\sim\mathcal{A}(C), s\sim p_C}[G_\tau(C) = 1, s \in O, \Theta(C,O) = 1] \end{aligned} \tag{37}$$

  By conditioning, we can decompose the last term of the previous equation into two terms

$$\begin{aligned} &\mathop{\mathbb{P}}_{C\sim\mathcal{D}, O\sim\mathcal{A}(C), s\sim p_C}[s \in O | G_\tau(C) = 1, \Theta(C,O) = 1] \\ &\mathop{\mathbb{P}}_{C\sim\mathcal{D}, O\sim\mathcal{A}(C), s\sim p_C}[G_\tau(C) = 1, \Theta(C,O) = 1] \end{aligned} \tag{38}$$

  Note that for the first term, we have

$$\mathop{\mathbb{P}}_{C\sim\mathcal{D}, O\sim\mathcal{A}(C), s\sim p_C}[s \in O | G_\tau(C) = 1, \Theta(C,O) = 1] \geq \frac{bk}{N} \tag{39}$$

since if $\Theta(C,O)=1$, the probability that $\mathbb{P}_{s\sim p_C}[s\in O]=\sum_{s'\in O}p_C(s')\geq bk/N$ by Equation 36. By the union bound inequality, we have that

$$\begin{aligned}\mathbb{P}_{C\sim\mathcal{D},O\sim\mathcal{A}(C),s\sim p_C}[G_\tau(C)=1,\Theta(C,O)=1]&\geq \mathbb{P}_{C\sim\mathcal{D}}[G_\tau(C)=1]+q-1\\&=p(\tau)+q-1\end{aligned}\tag{40}$$

since the probability that $\Theta(C,O)=1$ for any circuit C and string $O\sim\mathcal{A}(C)$ is at least q by assumption. Thus, by multiplying the terms in Equations 39,40 and combining it with Equation 37, we get that

$$\mu_1(\tau)\geq\frac{bk}{N}(p(\tau)+q-1)\tag{41}$$

Therefore, we have that

$$\frac{\mu_1(\tau)}{\mu_0(\tau)}\geq\frac{bk}{N}(p(\tau)+q-1)\cdot\frac{N}{bkp(\tau)}=\frac{p(\tau)+q-1}{p(\tau)}\tag{42}$$

as desired. ∎

It is easy to see that the lower bound for the ratio between the expectancies is monotonically non-increasing, as proven below.

Lemma 4 *The ratio $(p(\tau)+q-1)/p(\tau)$ is monotonically non-increasing for $\tau\in[0,1]$.*

Proof. Let $x,y\in[0,1], x\leq y$. Recall that $p(\tau)=\mathbb{P}_{C\sim\mathcal{D}}[\exists d:\mathbb{P}[\mathcal{A}(C)=d]\geq\tau/2^{B/2}]$, hence if $x\leq y\Rightarrow p(x)\geq p(y)$, since if there exists a d such that $\mathbb{P}[\mathcal{A}(C)=d]\geq y/2^{B/2}$, then $\mathbb{P}[\mathcal{A}(C)=d]\geq x/2^{B/2}$. Since q is a probability, note that $q-1\leq 0$, thus we have that

$$\begin{aligned}p(y)\leq p(x)\Rightarrow&(q-1)\cdot p(y)\geq(q-1)p(x)\\\Rightarrow&(q-1)\cdot p(y)+p(x)p(y)\geq(q-1)\cdot p(x)+p(x)p(y)\\\Rightarrow&p(y)(p(x)+q-1)\geq p(x)(p(y)+q-1)\\\Rightarrow&\frac{p(x)+q-1}{p(x)}\geq\frac{p(y)+q-1}{p(y)}\end{aligned}\tag{43}$$

as desired. ∎

We can now combine Lemmas 3 and 4 to obtain the following

$$\frac{\mu_1(\tau)}{\mu_0(\tau)}\geq\frac{\mu_1(1)}{\mu_0(1)}\geq b\cdot\frac{p(1)+q-1}{p(1)}=b\cdot\frac{p+q-1}{p}\geq 1+\epsilon\tag{44}$$

since, as previously discussed, $p(1)=p$.

Equation 44 allows us to conclude that the ratio between the expected number of output distributions that respect the assumption made in Equation 25 and the ones that don't exceeds one, and thus $\mu_1(\tau)-\mu_0(\tau)>0$, i.e., our assumption is indeed reasonable.

4.7 From Merlin-Arthur to Arthur-Merlin

In the previous subsection, we assumed the existence of a low-entropy device that solves the LXEB problem, and then proceeded to prove a series of Lemmas regarding such device. Equation 44 is almost what we need for us to use the QGSP in order to build an Arthur-Merlin protocol for the LLQSV, however, we do not yet know how to verify if Y_τ accepts an instance (C,s) in time $2^B n^{O(1)}$. The following Lemma aims to, for a fixed large enough T, find an index j that allows us to verify such condition efficiently.

Lemma 5 *Let $\mu_1(\tau), \mu_0(\tau)$ be defined according to Equation 29, and also assume that Equation 27 holds. Then for a given $T \geq \frac{8}{\epsilon}\log(\frac{N}{\epsilon}) : \exists j \in [T]$ such that*

$$\mu_1(1/2 + \frac{j}{T}) \geq (1 + \frac{\epsilon}{2})\mu_0(1/2 + \frac{j-1}{T}) \tag{45}$$

Proof. We prove the Lemma by contrapositive. Assume that

$$\forall j \in [T] : \mu_1(1/2 + \frac{j}{T}) < (1 + \frac{\epsilon}{2})\mu_0(1/2 + \frac{j-1}{T}) \tag{46}$$

We can consider the ratio $\mu_1(1/2)/\mu_1(1)$ and expand it into a telescoping product to obtain

$$\begin{aligned} \frac{\mu_1(1/2)}{\mu_1(1)} &= \prod_{j=1}^{T} \frac{\mu_1(1/2 + \frac{j}{T})}{\mu_1(1/2 + \frac{j+1}{T})} \\ &> (1 + \frac{\epsilon}{2})^{-T} \prod_{j=1}^{T} \frac{\mu_1(1/2 + \frac{j}{T})}{\mu_0(1/2 + \frac{j}{T})} \end{aligned} \tag{47}$$

Combining Lemma 4 with Equation 44 yields

$$\begin{aligned} \frac{\mu_1(1/2)}{\mu_1(1)} &> (\frac{1+\epsilon}{1+\epsilon/2})^T \\ &\geq (1 + \frac{\epsilon}{4})^T \end{aligned} \tag{48}$$

since $1 + \epsilon - (1 + \epsilon) - (1 + \epsilon/2)(1 + \epsilon/4) \geq 0$, for $\epsilon \leq 2$. We then let $T \geq \frac{8}{\epsilon}\log(\frac{N}{\epsilon})$, implying that

$$\begin{aligned} \mu_1(1/2) &> (1 + \frac{\epsilon}{4})^T \cdot \mu_1(1) \\ &\geq \frac{k}{N} \cdot \epsilon \cdot (1 + \frac{\epsilon}{4})^T \\ &\geq \frac{k}{N} \cdot \epsilon \cdot 2^{\epsilon\frac{T}{8}} \\ &\geq k \end{aligned} \tag{49}$$

but since $k \geq 1$, this implies that $\mu_1(1/2) > 1$, a contradiction, thus $T < \frac{8}{\epsilon}\log(\frac{N}{\epsilon})$, as desired. ∎

The previous Lemmas allows us to derive Merlin-Arthur protocol that verifies if $s \sim p_C$ in time $2^B \cdot T^2 \cdot n^{O(1)}$, which is described in Algorithm 3.

We now recall a useful result that will be used extensively on future proofs, and we shall state it without a proof.

Proposition 2 (Hoeffding's Inequality) *Let $X_1, X_2, ..., X_n$ be independent random variables such that $\mathbb{P}[a_i \leq X_i \leq b_i] = 1$, for real numbers a_i, b_i, and let $S_n = \sum_i X_i$. Then, $\forall t > 0$,*

$$\begin{aligned} \mathbb{P}[S_n - \mathbb{E}[S_n] \geq t] &\leq \exp(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}) \\ \mathbb{P}[|S_n - \mathbb{E}[S_n]| \geq t] &\leq 2\exp(-\frac{2t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}) \end{aligned} \tag{51}$$

Algorithm 3 Merlin-Arthur protocol for verifying if $s \sim p_C$
Input: circuit C, string $s \in \{0,1\}^n$,$K \in \mathbb{Z}$

Merlin sends a k-tuple $d = (z_1, ..., z_k)$ to **Arthur**, where each $z_i \in \{0,1\}^n$
Arthur makes $K+1$ samples of $\mathcal{A}(C)$, denoted as $d_1, ..., d_K, O$, where each sample is in turn a k-tuple of strings
Arthur accepts if

$$\tilde{\nu} \geq (\tau - \frac{1}{2T})2^{-B/2}, \tag{50}$$

where $\tilde{\nu} := \frac{1}{K}|\{l : d_l = d\}|$, and $s \in O$, and rejects otherwise.

We highlight the special case where $n = 1$, then:

$$\begin{aligned} \mathbb{P}[X - \mathbb{E}[X] \geq t] &\leq \exp(-\frac{2t^2}{(b-a)^2}) \\ \mathbb{P}[|X - \mathbb{E}[X]| \geq t] &\leq 2\exp(-\frac{2t^2}{(b-a)^2}) \end{aligned} \tag{52}$$

■

We are now extremely close to arriving at a contradiction: we just need to show that this Merlin-Arthur protocol does indeed solve the LLQSV problem in polynomial time. The following two Lemmas will prove precisely that.

Lemma 6 *For $\tau \in [1/2, 1], \eta = O(1/N^2), K \geq 4T^2 2^B n$, Algorithm 3 satisfies the following:*

1. *If $Y_\tau(C, s) = 1$, then Arthur accepts with probability $1 - \eta$.*
2. *If $Y_{\tau - 1/T}(C, s) = 0$, then Arthur rejects with probability $1 - \eta$.*

Proof. If $Y_\tau(C, s) = 1$, then by definition $\exists O = (z_1, ..., z_k) : \mathbb{P}[\mathcal{A}(C) = O] \geq \tau/2^{B/2}$, but since $\tau/2^{B/2} > \tau/2^B$, this also implies that there exists an $O = (z_1, ..., z_k) : \mathbb{P}[\mathcal{A}(C) = O] \geq \tau/2^B$. We can then use Hoeffding's inequality to show that

$$\mathbb{P}[|\tilde{\nu} - \mathbb{P}[\mathcal{A}(C) = O]| \geq \frac{1}{2T}2^{-B/2}] \leq 2\exp(-2\frac{K}{4T^2 2^B}) \leq 2\exp(-2n) \tag{53}$$

implying that Arthur rejects with probability at most $2\exp(-2n)$, and since $\eta = O(1/N^2)$, this means that he accepts with probability $1 - \eta$. If $Y_{\tau - 1/T}(C, s) = 0$, then for every O such that $s \in O$

$$\mathbb{P}[\mathcal{A}(C) = d] < \frac{\tau - 1/T}{2^{B/2}} \tag{54}$$

we can then apply Hoeffding's inequality again to obtain that Arthur accepts with probability at most $2\exp(-2n)$, and thus he rejects with probability $1 - \eta$. ■

We can now show that, for $M \geq N^3$, Algorithm 3 accepts that at least K tuples from the *Yes-case* of LLQSV, and accepts at most $(1 - \Omega(\epsilon))K$ tuples from the *No-case* of LLQSV, both with probability $1 - 2^{-\Omega(n)}$, i.e., that such algorithm satifies the condition for using the QGSP.

Lemma 7 *Let $M \geq N^3, \eta' = 2^{-\Omega(N)}$. Then $\exists K \in [M]$ such that, wich probability at least $1 - \eta'$, Arthur accepts at least K tuples in an Yes-case of the LLQSV, and accepts at most $K' = (1 - \epsilon/5 + O(\epsilon^2))K$ tuples in a No-case of the LLQSV.*

Proof. Recall that $\mu_1(\tau) = \mathop{\mathbb{E}}_{C\sim\mathcal{D},s\sim p_C}[Y_\tau(C,s)]$, and that $\mu_0(\tau) = \mathop{\mathbb{E}}_{C\sim\mathcal{D},s\sim\mathcal{U}}[Y_\tau(C,s)]$. Note that any given tuple, both in the Yes-case and the No-case of the LLQSV, is sampled independently, thus, by Lemma 6, Arthur accepts each sample with probability at most $\mu_0(\tau)+\eta$. We can then set $K' = (\mu_0(\tau)+\eta)(1+\epsilon/8)M$, and conclude using Hoeffding's inequality that

$$\begin{aligned}\mathop{\mathbb{P}}_{C_1,\ldots,C_M\sim\mathcal{D},s_1,\ldots,s_m\sim\mathcal{U}}[\text{Arthur accepts more than } K' \text{ tuples}] &\leq 2\exp(\frac{-M\epsilon^2(\mu_0(\tau)+\eta)^2}{32}) \\ &= 2\exp(-\Omega(\frac{M\epsilon^2k^2p^2}{N^2})) \\ &\leq 2\exp(-\Omega(\frac{M\epsilon^4k^2}{N^2}))\end{aligned} \tag{55}$$

since, according to Equation 27, $\mu_0(\tau) = p(\tau)k/N \geq pk/N \geq \epsilon k/N$. Since we assume that $M \geq N^3$, this yields that

$$\mathop{\mathbb{P}}_{C_1,\ldots,C_M\sim\mathcal{D},s_1,\ldots,s_m\sim\mathcal{U}}[\text{Arthur accepts more than } K' \text{ tuples}] \leq 2\exp(-\Omega(N)) \tag{56}$$

implying that Arthur accepts at most K' tuples with probability at least $1-2\exp(-\Omega(N)) \geq 1-\eta'$. Now for the Yes-case, we can take $K = (\mu_1(\tau)-\eta)(1-\epsilon/8)M$, and conclude again by Hoeffding's inequality that

$$\begin{aligned}\mathop{\mathbb{P}}_{C_1,\ldots,C_M\sim\mathcal{D},s_i\sim p_{C_i}}[\text{Arthur accepts fewer than } K \text{ tuples}] &\leq 2\exp(\frac{-M\epsilon^2(\mu_1(\tau)-\eta)^2}{32}) \\ &\leq 2\exp(-\Omega(\frac{M\epsilon^2k^2p^2}{N^2}))\end{aligned} \tag{57}$$

since $\eta = O(1/N^2), \mu_1(\tau)-\eta = \Omega(\mu_1(\tau)) = \Omega(\mu_0(\tau)) = \Omega(pk/N)$, which in turn implies that Arthur accepts at least K tuples in the Yes-case with probability at least $1-\eta'$. We now need to show that $K'/K = (1-\epsilon/5+O(\epsilon^2))$, but since $\eta = o(\epsilon)$, we have that

$$\begin{aligned}\frac{K'}{K} &\leq \frac{\mu_0(\tau)+\eta}{\mu_1(\tau)-\eta}\cdot\frac{1+\epsilon/8}{1-\epsilon/8} \\ &\leq \frac{\mu_0(\tau)+\eta}{\mu_1(\tau)-\eta}\cdot\frac{1}{1-\epsilon/4} \\ &\leq \frac{1}{1+\epsilon/4-O(\eta)} \\ &\leq \frac{1}{1+\epsilon/5} \\ &\leq 1-\epsilon/5+O(\epsilon^2)\end{aligned} \tag{58}$$

as desired. ∎

Thus, we have shown that our low-entropy device solving LXEB does indeed give birth to a Merlin-Arthur protocol for the LLQSV problem that satisfies the necessary conditions for the QGSP, stated in subsection 4.5. We can now provide a full description of the QGSP derived from our low-entropy device, which is described in Algorithm 4.

Algorithm 4 Quantum Goldwasser-Sipser Protocol for the LLQSV problem, using a low-entropy device that solves the LXEB

Input: both Merlin and Arthur are given access to $(C_1, s_1), ..., (C_M, s_M)$, and integers $T, j \in [T], R$.

Arthur chooses a random hash function $f : [M] \mapsto [R]$, and uniformly chooses $y \in [R]$

Merlin sends an integer i and $d = (z_1, ..., z_k)$ to **Arthur**

Given $K \geq 4T^2 2^B n$, Arthur takes $K + 1$ samples $d_1, ..., d_k, O \sim \mathcal{C}$. He accepts if

(a) $\tilde{\nu} \geq (\tau - \frac{1}{2T})2^{-B/2}$, where $\tilde{\nu} = \frac{1}{K}|\{l : d_l = d\}|, \tau = 1/2 + j/T$,

(b) $s_i \in O$, and

(c) $f(i) = y$

4.8 Arriving at a contradiction

We can now finally prove the main result of Section 5 from [AH23].

Theorem 1 *If there exists a quantum device $\mathcal{A}$ which runs in quantum polynomial time and that satisfies Assumption 2, then LLQSV$(\mathcal{D}) \in$ QCAMTIME$(2^B n^{O(1)})/q(2^B n^{O(1)})$, i.e., the LLHA is false.*

Proof. The idea of this proof is to show that the aforementioned protocol solves the LLQSV$(\mathcal{D})$ problem with constant gap. Algorithm 4 describes the QGSP version of the low-entropy device $\mathcal{A}$, and by Lemma 7 we know that $K' \leq (1-\epsilon/6)K$. We can then employ Lemma 2 to Algorithm 4 to conclude that the protocol's gap is at least $\epsilon^2/4\alpha$, where $\alpha > 1$, which in turn implies that the gap is $\Omega(\epsilon)$. Therefore, if we run a $(1/\epsilon)^{O(1)}$-fold parallel repetition of Algorithm 4, we obtain a protocol with constant gap for the LLQSV$_B(\mathcal{D})$ problem. This in turn implies that Algorithm 4 solves the problem in $2^B n^{O(1)}$ time, with advice string of size $2^B n^{O(1)}$, and so we conclude that LLQSV$_B(\mathcal{D}) \in$ QCAMTIME$(2^B n^{O(1)})/$q$(2^B n^{O(1)})$. ∎

We first assumed that the LLHA is true, however the previous theorem clearly contradicts it, and thus Assumption 2 must be false, implying that there can be no low-entropy quantum device $\mathcal{A}$ solving the LXEB problem under the LLHA. Moreover, we can also conclude that, if the LLHA holds, then the negation of Equation 27 must hold, i.e.,

$$p \leq \frac{b}{b-1}(1-q) + n^{-\omega(1)}. \tag{59}$$

which in turn yields the following corollary.

Corollary 1 *If the LLHA$(\mathcal{D})$ holds for a distribution $\mathcal{D}$ over $U(N)$, then every quantum device $\mathcal{A}$ solving LXEB(b, k) with probability q over choices of $C \sim \mathcal{D}$ satisfies*

$$\mathbb{P}_{C\sim\mathcal{D}}[H_{min}(\mathcal{A}(C)) \geq B/2] \geq \frac{bq-1}{b-1} - n^{-\omega(1)} \tag{60}$$

In other words, the output of $\mathcal{A}$ contains $\Omega(n)$ pure random bits. ∎

The previous corollary concludes the proof. We can thus affirm that, if the LLHA holds, the protocol described in subsection 4.1 does indeed work.

4.9 A QRNG protocol with certified randomness

Finally, we provide a full description of the protocol conceived by Aaronson and Hung for generating cryptographically certified random bits from a process similar to the RCS experiments

described in subsection 4.1. We describe the protocol in Algorithm 5.

Algorithm 5 Aaronson and Hung's cryptographically certified Quantum Random Number Generator

Input: number of qubits n, a distribution $\mathcal{D}$ over $U(2^n)$, the constants $b \in (1,2), k = O(n^2)$ for computing LXEB(b,k), the number of rounds m, and a parameter $\gamma = O(\frac{\log(n)}{m})$ controlling the number of circuit updates.

for $i \in [m]$ **do**

The **user** samples $T_i \sim$ Bernoulli(γ), where T_i will be a variable controlling if the current round i will be a testing round

if $(i > 1 \wedge T_{i-1} = 1) \vee i = 1$ **then**

The **user** samples $C_i \sim \mathcal{D}$

else

The **user** sets $C_i = C_{i-1}$

end if

The **user** sends C_i to the **server**, and keeps T_i secret

The **server** returns k samples $d_i = (z_1, ..., z_k)$

If $T_i = 1$, the **user** sets

$$W_i := \begin{cases} 1, & \text{if } F_{\text{LXEB}(b,k)}(z_1, ..., z_k) \geq 0 \text{ and } E_i = 1 \\ 0, & \text{otherwise} \end{cases} \tag{61}$$

where $E_i = 0$ if there exists distinct $l, l' \in \{j : C_j = C_i\}$ such that the samples $d_l, d_{l'}$ are not all distinct (this is used to prevent the **server** repeating responses for any two rounds using the same challenge circuit). If $T_i = 0$, the **user** sets $W_i = 0$

end for

Define $t = |\{i : T_i = 1\}|$ as the number of test rounds. The **user** then computes

$$W = \sum_{i:T_i=1} W_i \tag{62}$$

if $W \geq 0.99t$ **then**

The **user** accepts the **server**'s output, and uses $(d_1, ..., d_m)$ as an input for a randomness extractor

else

The **user** rejects

end if

5 Conclusion

In this work, we provide an extensive overview of the topic of Quantum Random Number Generation and its applications to contemporary computer science, physics and mathematics. We first introduce the necessary concepts from classical RNG, classical and quantum information theory, and quantum complexity theory that are required for fully grasping the subject. We then proceed to discuss the notion of Quantum Randomness and formally define the concept of a QRNG.

Finally, we discuss the recent findings of Scott Aaronson and Shih-Han Hung in [AH23].

More specifically, we provide a full proof of one of their main results: given certain reasonable assumptions regarding the hardness of the Long List Quantum Supremacy Verification task with respect to the Quantum Classical Arthur-Merlin complexity class, then any quantum device solving the Linear Cross-Entropy Benchmarking problem in polynomial time must output close to n nearly pure random bits, where n is the number of qubits of the system. This in turn allows for the creation of a QRNG protocol with a good lower bound on the number of random bits in its output, and that also intrinsically require the use of a quantum computer. Moreover, this protocol is designed in the context of the recent Quantum Supremacy experiments made by Google and USTC, and so it can be implemented with current technology.

Overall, our work provides an in-depth analysis of some of the most relevant theoretical and practical aspects of QRNG available today, while at the same time trying to be accessible to most readers who come from a STEM background, but who might not be familiar with QRNG, or with quantum mechanics as a whole. Therefore, this article may serve as a good first step for any such reader interested in starting to learn about the applications of quantum mechanics to random number generation in a more introductory yet comprehensive way.

References

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, Oct 2019.

[AGL+23] Dorit Aharonov, Xun Gao, Zeph Landau, Yunchao Liu, and Umesh Vazirani. A polynomial-time classical algorithm for noisy random circuit sampling. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. ACM, jun 2023.

[AH23] Scott Aaronson and Shih-Han Hung. Certified randomness from quantum supremacy, 2023.

[Bab85] L Babai. Trading group theory for randomness. In *Proceedings of the Seventeenth*

Annual ACM Symposium on Theory of Computing, STOC '85, page 421–429, New York, NY, USA, 1985. Association for Computing Machinery.

[Cha69] G. J. Chaitin. On the simplicity and speed of programs for computing infinite sets of natural numbers, 1969.

[Fei15] Joan Feigenbaum. Lecture notes in computational complexity. https://zoo.cs.yale.edu/classes/cs468/previous-years/spr15/, Spring 2015.

[GKC+21] Xun Gao, Marcin Kalinowski, Chi-Ning Chou, Mikhail D. Lukin, Boaz Barak, and Soonwon Choi. Limitations of linear cross-entropy as a measure for quantum advantage, 2021.

[GS86] S Goldwasser and M Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, page 59–68, New York, NY, USA, 1986. Association for Computing Machinery.

[JJNJ21] Marcin M. Jacak, Piotr Jóźwiak, Jakub Niemczuk, and Janusz E. Jacak. Quantum generators of random numbers. *Scientific Reports*, 11(1):16108, Aug 2021.

[Khr15] Andrei Khrennikov. Randomness: quantum versus classical, 2015.

[Knu97] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Boston, third edition, 1997.

[Kol63] A. N. Kolmolgorov. On tables of random numbers, 1963.

[Kol98] A. N. Kolmolgorov. On tables of random numbers, 1998.

[L'E17] Pierre L'Ecuyer. History of uniform random number generation. In *Proceedings of the 2017 Winter Simulation Conference*, WSC '17. IEEE Press, 2017.

[Mac03] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Copyright Cambridge University Press, 2003.

[Mar03] George Marsaglia. Xorshift rngs. *Journal of Statistical Software*, 8(14):1–6, 2003. https://pt.wikipedia.org/wiki/Xorshift.

[Mic23] MicrosoftResearch. Z3: The z3 theorem prover, 2023. https://github.com/Z3Prover/z3.

[MYC+16] Xiongfeng Ma, Xiao Yuan, Zhu Cao, Bing Qi, and Zhen Zhang. Quantum random number generation. *npj Quantum Information*, 2(1):16021, Jun 2016.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[NRK+18] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, A. Megrant, B. Chiaro, A. Dunsworth, K. Arya, R. Barends, B. Burkett, Y. Chen, Z. Chen, A. Fowler, B. Foxen, M. Giustina, R. Graff, E. Jeffrey, T. Huang, J. Kelly, P. Klimov, E. Lucero, J. Mutus, M. Neeley, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. C. White, H. Neven, and J. M. Martinis. A blueprint for demonstrating quantum supremacy with superconducting qubits. *Science*, 360(6385):195–199, apr 2018.

[TS21] K. Tamura and Y. Shikano. *Quantum Random Numbers Generated by a Cloud Superconducting Quantum Computer*, volume 33 of *Mathematics for Industry*. Springer, Singapore, 2021.

[Wik] Wikipedia. Fortuna algorithm - wikipedia. https://en.wikipedia.org/wiki/Fortuna_(PRNG). [Online; accessed 2-July-2023].

[Wik23a] Wikipedia. Arthur–Merlin protocol — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Arthur%E2%80%93Merlin%20protocol&oldid=1159310244, 2023. [Online; accessed 16-June-2023].

[Wik23b] Wikipedia. Interactive proof system — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Interactive%20proof%20system&oldid=1158196616, 2023. [Online; accessed 19-June-2023].

[Wik23c] Wikipedia. Min-entropy — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Min-entropy&oldid=1112107993, 2023. [Online; accessed 26-June-2023].