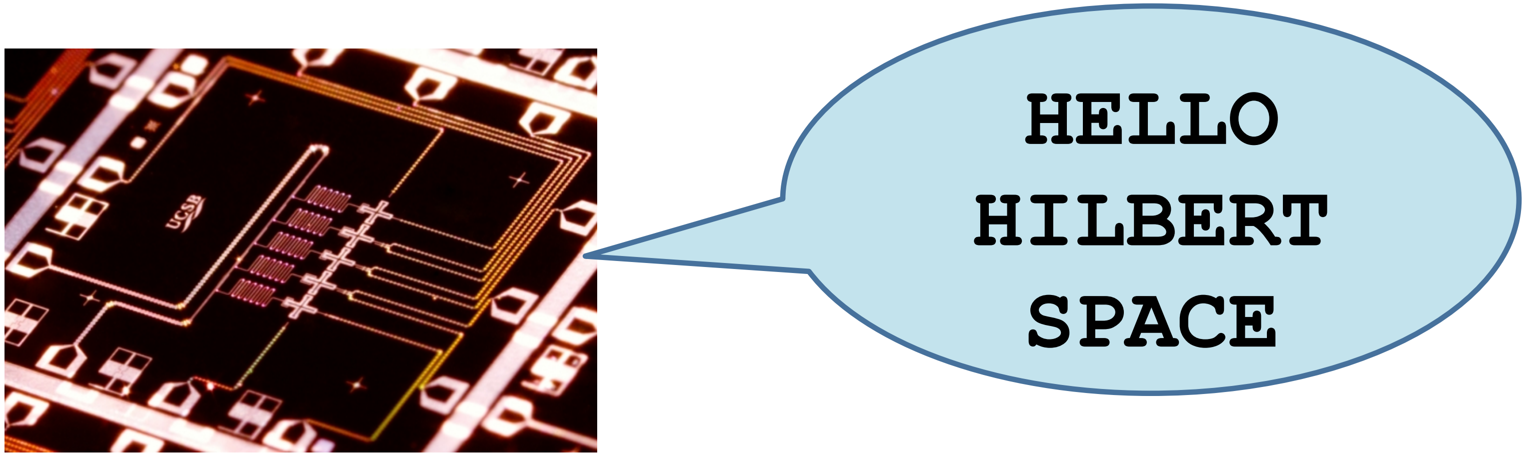
Quantum Supremacy and its Applications



Scott Aaronson (University of Texas, Austin)

Simons Institute, Berkeley, June 12, 2018

Based on joint work with Lijie Chen (CCC'2017, arXiv: 1612.05903) and on forthcoming work

Papers and slides at www.scottaaronson.com

QUANTUM SUPREMACY

#1 Application of quantum computing: Disprove Gil Kalai! (And the Extended Church-Turing Thesis)

Might actually be able to achieve in the next couple years, e.g. with Google's 72-qubit Bristlecone chip. "Obviously useless for anything else," but who cares?

The Sampling Approach

Put forward by A.-Arkhipov 2011 (BosonSampling), Bremner-Jozsa-Shepherd 2011 (IQP Model), and others

Consider problems where the goal is to **sample** from a desired distribution over n-bit strings

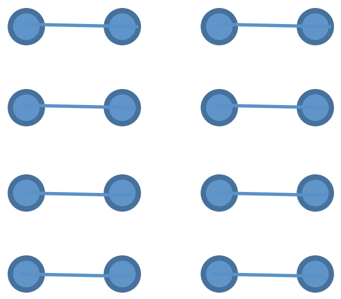
Compared to problems with a single valid output (like FACTORING), sampling problems can be

(1) Easier to solve with near-future quantum devices, and

(2) Easier to argue are hard for classical computers!

(We "merely" give up on: practical applications, fast classical way to verify the result…?)

The Random Quantum Circuit Proposal



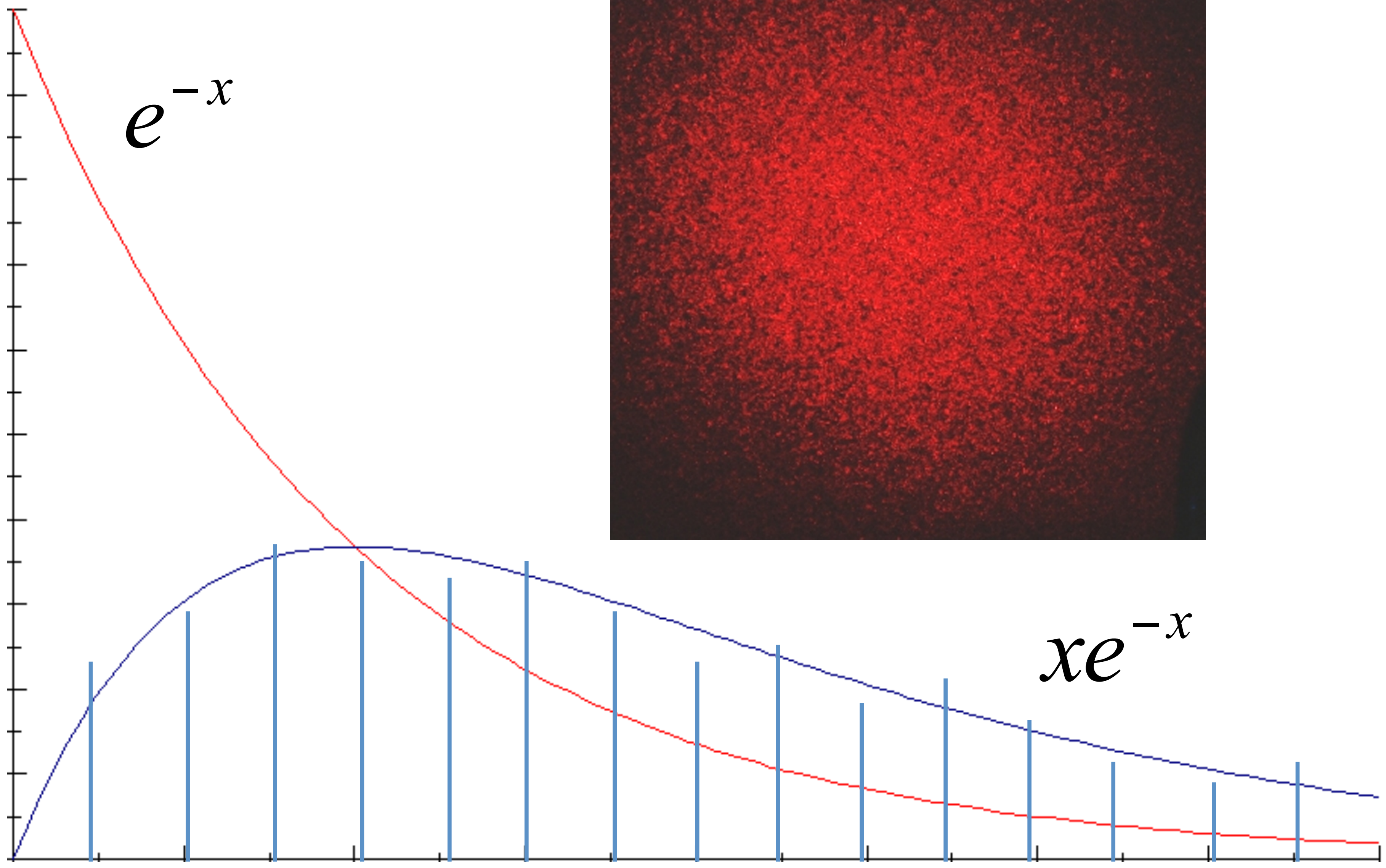
Generate a quantum circuit C on n qubits in a $\sqrt{n}\times\sqrt{n}$ lattice, with d layers of random nearest-neighbor gates

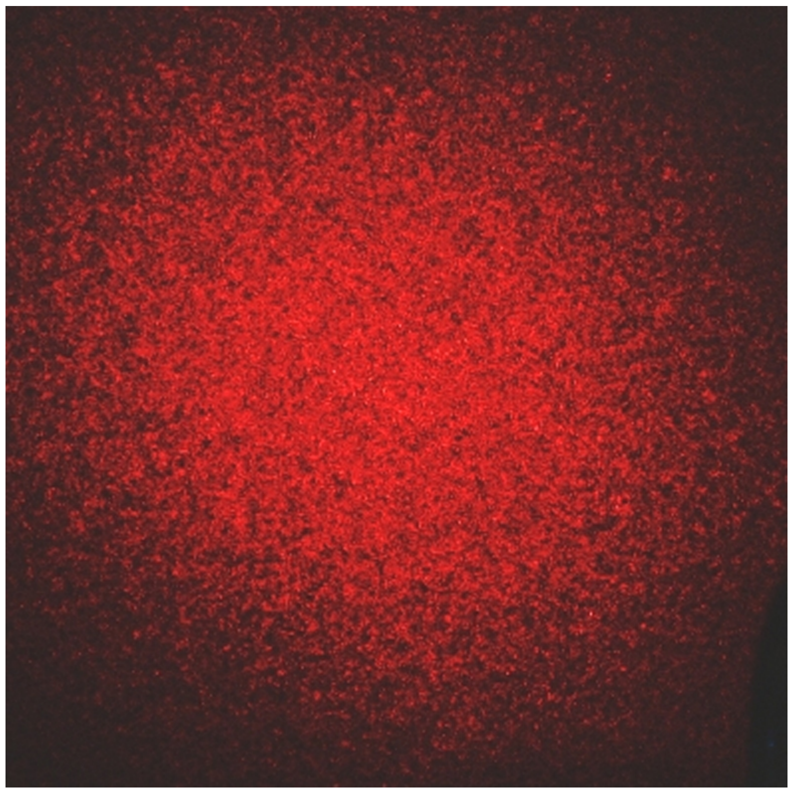
Apply C to $|0\rangle^{\otimes n}$ and measure. Repeat T times, to obtain samples $x_1,\ldots,x_T$ from $\{0,1\}^n$

Check whether $x_1,\ldots,x_T$ solve the “Heavy Output Generation” (HOG) problem—e.g., do at least 2/3 of the x_i’s have more than the median probability?

(takes classical exponential time, which is OK for $n\approx 70$)

Publish C. Challenge skeptics to generate samples passing the test in a reasonable amount of time


e^{-x}
xe^{-x}



Our Strong Hardness Assumption

There's no polynomial-time classical algorithm A such that, given a uniformly-random quantum circuit C with n qubits and m>>n gates,

$$\Pr_C\left[A(C)\text{ guesses whether }\left|\langle 0|^{\otimes n} C|0\rangle^{\otimes n}\right|^2 > \text{median}\right] \geq \frac{1}{2} + \Omega\left(2^{-n}\right)$$

Note: There *is* a polynomial-time classical algorithm that guesses with probability $\approx \frac{1}{2} + \frac{1}{4^m}$

(just expand $\langle 0|^{\otimes n}C|0\rangle^{\otimes n}$ out as a sum of 4^m terms, then sample a few random ones)

Theorem: Assume SHA. Then given as input a random quantum circuit C, with n qubits and m>>n gates, there's no polynomial-time classical algorithm that even **passes our statistical test for C-sampling** w.h.p.

Pro e
we
out $\otimes n$

Of course we'd like hardness of random circuit sampling based on a weaker complexity assumption. Recent partial progress in that direction by Bouland, Fefferman, Nirkhe, Vazirani arXiv:1803.04402

Nov
test for C' with probability ≥0.99. Suppose A outputs samples $x_1,\ldots,x_T$. Then if $x_i = z$ for some $i \in [T]$, guess that

$$\left|\langle 0|^{\otimes n} C |0\rangle^{\otimes n}\right|^2 \geq \text{median}$$

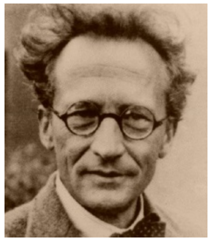
Otherwise, guess that with probability $\frac{1}{2} - \frac{T}{2^{n+1}}$

Violates SHA!

Time-Space Tradeoffs for Simulating Quantum Circuits

Given a general quantum circuit with n qubits and m>>n two-qubit gates, how should we simulate it classically?

"Schrödinger way":

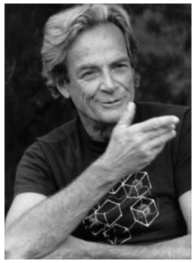


Store whole wavefunction

$O(2^n)$ memory, $O(m2^n)$ time

n=40, m=1000: Feasible but requires TB of RAM

"Feynman way":



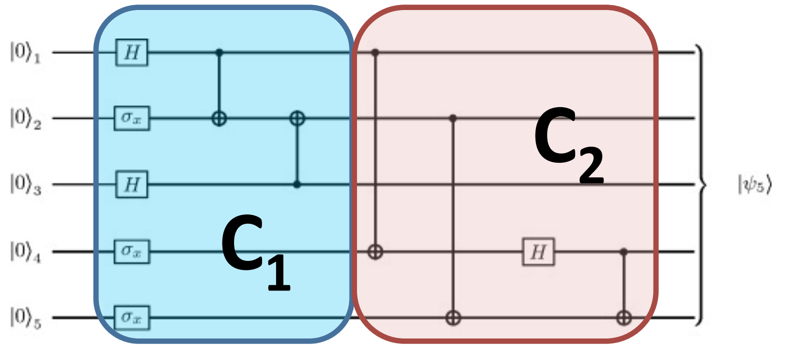
Sum over paths

$O(m+n)$ memory, $O(4^m)$ time

n=40, m=1000: Infeasible but requires little RAM

Best of both worlds?

Theorem: Let C be a quantum circuit with n qubits and d layers of gates. Then we can compute each transition amplitude, $\langle x|C|y\rangle$, in $d^{O(n)}$ time and poly(n,d) memory



Proof: Savitch's Theorem! Recursively divide C into two chunks, C_1 and C_2, with d/2 layers each. Then

$$\langle x|C|y\rangle = \sum_{z\in\{0,1\}^n} \langle x|C_1|z\rangle\langle z|C_2|y\rangle$$

Can do better for nearest-neighbor circuits, or when more memory is available

This algorithm still doesn't falsify the SHA! Why not?

What About Errors?

k bit-flip errors ⇒ deviation from the uniform distribution is suppressed by a 1/exp(k) factor. Without error-correction, can only tolerate a few errors. Will come down to numbers.

Verification

Needs to be difficult but not impossible (like Bitcoin mining). Partly using our recursive approach, Pednault et al. from IBM and Chen et al. from Alibaba have now shown how to handle up to ~70 qubits classically. Perfectly consistent with what we're trying to do!

Random Bits are *Obviously* Useless…

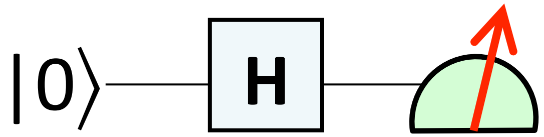
11010000110100111101101100110011000101001001

Certified Random Bits: Who Needs 'Em?

For private use:
Cryptographic keys (a big one!)

For public use:
Election auditing, lotteries, parameters for cryptosystems, zero-knowledge protocols, proof-of-stake cryptocurrencies…

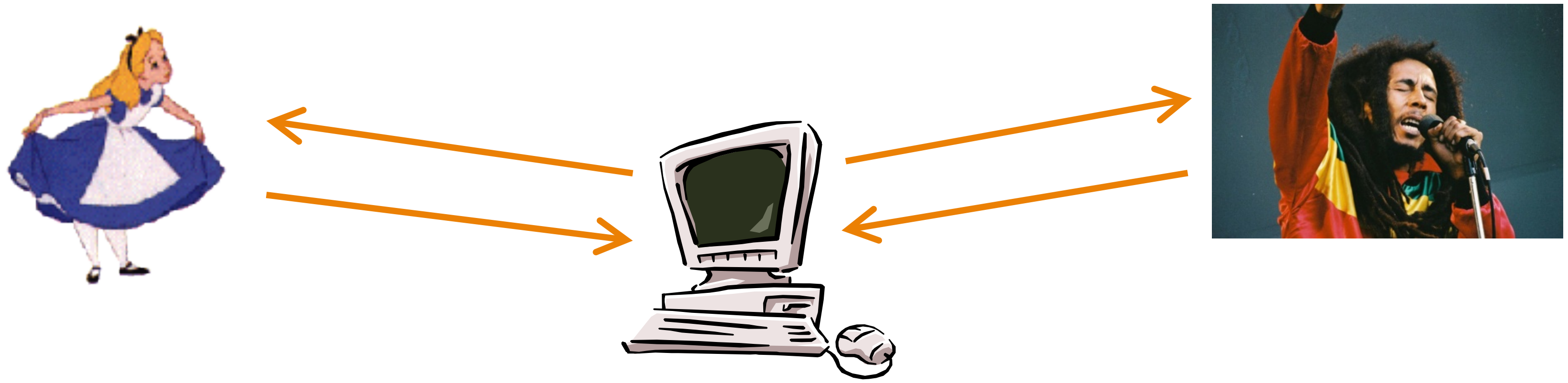
Trivial Quantum Randomness Solution!



Problem: What if your quantum hardware was backdoored by the NSA? (Like the DUAL_EC_DRBG pseudorandom generator was?) Want to trust a deterministic classical computer only

Earlier Approach: Bell-Certified Randomness Generation

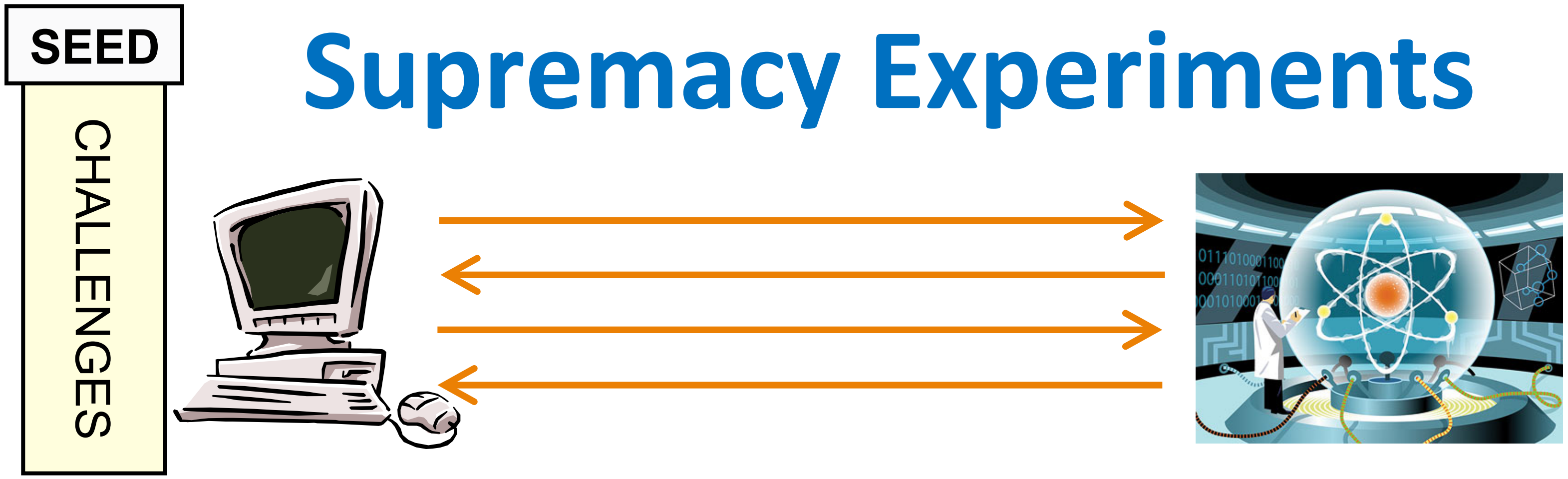
Colbeck and Renner, Pironio et al., Vazirani and Vidick, Coudron and Yuen, Miller and Shi…



Upside: Doesn't need a QC; uses only "current technology" (though loophole-free Bell violations are only ~2 years old)

Downside: If you're getting the random bits over the

Randomness from Quantum Supremacy Experiments

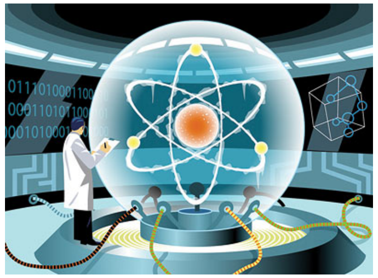


Key Insight: A QC can solve certain sampling problems quickly—but under plausible hardness assumptions, it can **only** do so by sampling (and hence, generating real entropy)

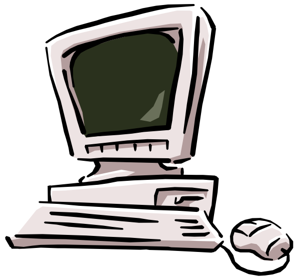
Upsides: Requires just a single device—good for certified randomness over the Internet. Ideally suited to NISQ devices

Caveats: Requires hardness assumptions and initial seed randomness. Verification (with my scheme) takes exp(n)

Applications



For the QC owner: Private randomness



For those connecting over the cloud: Public randomness

The protocol does require pseudorandom challenges, but:

Even if the pseudorandom generator is broken later, the truly random bits will remain safe (“forward secrecy”)

Even if the seed was public, the random bits can be private

The random bits demonstrably weren’t known to *anyone*, even the QC, before it received a challenge (freshness)

The Protocol

1. The classical client generates n-qubit quantum circuits C_1, …,C_T pseudorandomly (mimicking a random ensemble)

2. For each t, the client sends C_t to the server, then demands a response S_t within a very short time

    In the "honest" case, the response is a list of k samples from the output distribution of $C_t|0\rangle^{\otimes n}$

3. The client picks O(1) random iterations t, and for each one, checks whether S_t solves "HOG" (Heavy Output Generation)

4. If these checks pass, then the client feeds $S=\langle S_1,\ldots,S_T\rangle$ into a classical **randomness extractor**, such as GUV (Guruswami-Umans-Vadhan), to get nearly pure random bits

Main Result

Suppose that suitable hardness assumptions hold, and that the server does at most $n^{O(1)}$ quantum computation per iteration. Suppose also that we run the protocol, for $T \leq 2^n$ steps, and the client accepts with probability >½. Then conditioned on the client accepting, the output bits S are $1/\exp(n^{\Omega(1)})$-close in variation distance to a distribution with **min-entropy** $\Omega(Tn)$.

$$H_{\min}(D) := \min_x \log_2 \frac{1}{\Pr_D[x]}$$

Which means: the extractor will output $\Omega(Tn)$ bits that are exponentially close to uniform

Hardest part: show **accumulation** of min-entropy across the T iterations. E.g., rule out that the samples are correlated

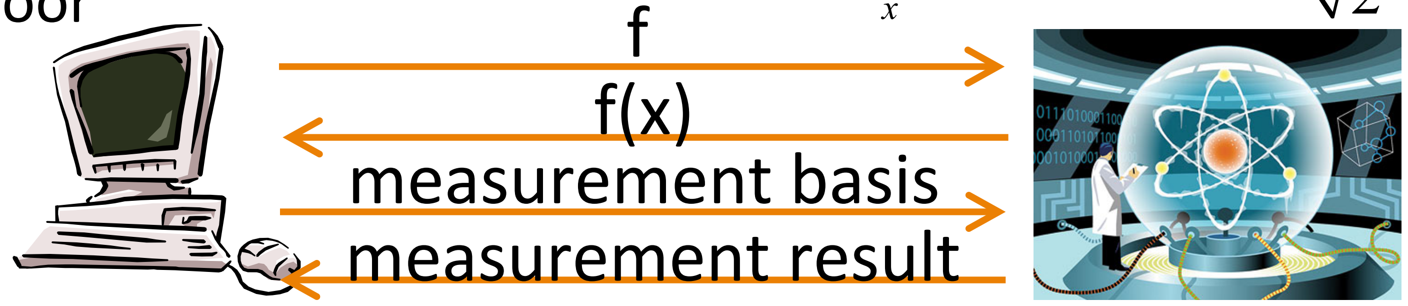
Different Approach

Brakerski, Christiano, Mahadev, Vazirani, Vidick arXiv:1804.00640

Method for a QC to generate random bits, assuming the quantum hardness of breaking lattice-based cryptosystems

2-to-1 function f,
plus trapdoor

$$\sum_x |x\rangle |f(x)\rangle \to \frac{|x\rangle + |y\rangle}{\sqrt{2}}$$



Huge advantage of the BCMVV scheme over mine: Polynomial-time classical verification!

Advantage of mine: Can be run on NISQ devices!

Future Directions

Can we get quantum supremacy, as well as certified randomness, under more “standard” and less “boutique” complexity assumptions?

Can we get polynomial-time classical verification and NISQ implementability at the same time?

Can we get more and more certified randomness by sampling with the **same** circuit C over and over? Would greatly improve the bit rate, remove the need for a PRF

Can we prove our randomness scheme sound even against adversaries that are entangled with the QC?

Conclusions

We might be close to ~70-qubit quantum supremacy experiments. We can say nontrivial things about the hardness of simulating these experiments, but we'd like to say more

Certified randomness generation: the **most** plausible application of very-near-term QCs?

This application **requires** sampling problems: problems with definite answers (like factoring) are useless!

Not only can we do it with ~70 qubits, we don't **want** more. No expensive encoding needed; can fully exploit hardware

With this application, all the weaknesses of sampling-based quantum supremacy experiments become strengths!