Editors' Suggestion

Experimental implementation of an efficient test of quantumness

Laura Lewis,[1,2,*] Daiwei Zhu,[3,4,5,6] Alexandru Gheorghiu,[7] Crystal Noel,[3,8,9] Or Katz,[8,9] Bahaa Harraz,[3] Qingfeng Wang,[3,4,10] Andrew Risinger,[3,4] Lei Feng,[3,4] Debopriyo Biswas,[3,4] Laird Egan,[3,4] Thomas Vidick,[1] Marko Cetina,[3,8] and Christopher Monroe[3,4,5,8,9]

[1]*Institute for Quantum Information and Matter and Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, California 91125, USA*

[2]*Division of Physics, Mathematics, and Astronomy, California Institute of Technology, Pasadena, California 91125, USA*

[3]*Joint Quantum Institute, Departments of Physics and Electrical and Computer Engineering, University of Maryland, College Park, Maryland 20742, USA*

[4]*Joint Center for Quantum Information and Computer Science, NIST/University of Maryland, College Park, Maryland 20742, USA*

[5]*IonQ, Inc., College Park, Maryland 20740, USA*

[6]*Departments of Electrical and Computer Engineering, University of Maryland, College Park, Maryland 20742, USA*

[7]*Institute for Theoretical Studies, ETH Zürich, Zürich CH 8001, Switzerland*

[8]*Duke Quantum Center and Department of Physics, Duke University, Durham, North Carolina 27708, USA*

[9]*Department of Electrical and Computer Engineering, Duke University, Durham, North Carolina 27708, USA*

[10]*Chemical Physics Program and Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742, USA*

(Received 16 October 2022; revised 25 October 2023; accepted 2 November 2023; published 9 January 2024)

A test of quantumness is a protocol where a classical user issues challenges to a quantum device to determine if it exhibits nonclassical behavior, under certain cryptographic assumptions. Recent attempts to implement such tests on current quantum computers rely on either interactive challenges with efficient verification or noninteractive challenges with inefficient (exponential time) verification. In this paper, we execute an efficient noninteractive test of quantumness on an ion-trap quantum computer. Our results significantly exceed the bound for a classical device's success.

DOI: 10.1103/PhysRevA.109.012610

I. INTRODUCTION

As research in quantum theory continues to advance, experimentally testing the validity of the theory becomes of greater importance. In particular, a key question is whether quantum mechanics is falsifiable in the regime of high complexity arising from large entangled states [1]. This is exceptionally difficult to answer due to the exponential complexity in representing general quantum systems. Traditionally, one can test a physical theory by first predicting an outcome according to the theory and comparing with the experimental result. In quantum mechanics, such predictions can require exponential resources to obtain and therefore do not provide a feasible approach for validating the theory.

The work of [1] proposed one way to overcome this exponential overhead: using interactive protocols known as *interactive proof systems* [2–6]. Such protocols allow a computationally weak *verifier* to test the behavior of a powerful *prover* (or even multiple provers [7–10]). The protocols work by having the verifier issue a challenge to the prover, the prover responds, the verifier issues another challenge, and the process repeats. After a certain number of rounds, the verifier either accepts or rejects based on the prover's responses in all rounds. Interactive proofs have been key to several developments in complexity theory and cryptography [4,11–13].

For the specific case in which the prover is a quantum computer, a number of protocols have been proposed to verify the results of general quantum computations [14]. Prior to 2018, all such protocols required quantum communication between the verifier and the prover. This changed following the breakthrough result of Mahadev, who gave the first protocol for quantum verification using only a classical verifier [15]. At the same time, Brakerski *et al.* introduced the concept of a *test (or proof) of quantumness* [16]. This is a protocol in which the verifier simply wishes to determine if the prover is nonclassical. In other words, a test of quantumness is an interactive protocol in which an efficient quantum prover can make the verifier accept (with high probability) and no efficient classical prover can make the verifier accept (with high probability). "Efficient" in this context means that the prover runs in polynomial time. The challenges issued by the verifier in this protocol are constructed so that a classical prover would be unable to answer them, unless it is able to efficiently solve hard cryptographic problems [such as factoring, or the learning with errors (LWE) problem [17]]. On the other hand, the quantum prover is able to answer these challenges, without necessarily violating the intractability of the cryptographic tasks. Crucially, the verifier can efficiently check whether the challenges were answered correctly or not. This then serves as a test of quantum behavior under certain cryptographic assumptions.

Several recent works have addressed the problem of constructing cryptographic proofs of quantumness [16,18–23]. As these are interactive protocols, the main challenge towards

*llewis@caltech.edu

an experimental implementation is that the quantum prover must perform *mid-circuit measurements* in order to correctly answer the verifier's challenges. While extensive research has been conducted on how to effectively implement mid-circuit measurements [24–26], the process introduces more errors compared to performing terminal measurements, which occur only at the end of the algorithm. The feasibility of implementing interactive cryptographic proofs of quantumness via mid-circuit measurements with near-term devices was recently demonstrated in [22]. There, it was observed that when implementing both mid-circuit measurements and terminal measurements for the same protocol, the protocols with intermediate measurements resulted in a significantly lower success probability. Moreover, other currently known interactive protocols rely on heavy cryptographic assumptions that have a large impact on the qubit count and depth of the required circuits. It would therefore be desirable to have a test of quantumness that is *noninteractive*.

Such a protocol was proposed in [27]. This replaces the need for interaction with the use of a one-bit *hash function*. The high-level idea is that because the hash function acts as a random function (or, more formally, as a *random oracle*), in order to succeed in the verifier's new challenge involving the hash function, the prover must effectively have been able to answer both branches of the interactive version of the protocol. In a sense, the hash function accounts for both branches of the interactive protocol, eliminating the need for interaction. The idea of using hash functions to eliminate interaction originates in cryptography, where it is known as the Fiat-Shamir heuristic [28].

This technique opens up more possibilities for efficient tests of quantum mechanics on near-term devices and contrasts the approaches used previously to certify quantum advantage [29,30]. Those approaches are based on delegating a sampling task to the quantum device (such as random circuit sampling or boson sampling) and then checking the validity of the obtained samples using the *linear cross-entropy benchmark (LXEB)* [31]. The major downside of this approach is that computing the LXEB takes exponential time, meaning that certifying quantum advantage in this way quickly becomes intractable [29,31,32]. In addition, there are situations in which the LXEB can be "classically spoofed" (i.e., there is an efficient classical algorithm which can produce samples that are valid according to the LXEB) [33,34]. The proof of quantumness of [27], on the other hand, requires only polynomial runtime to perform the certification and is thus efficient. In addition, classically spoofing the results of a proof of quantumness is (provably) as hard as breaking the underlying cryptography (this follows from the soundness of these protocols against classical provers as shown in [16,27]). We note that a new noninteractive test of quantumness was recently introduced in [35], which only relies on one of the two cryptographic assumptions required in [27] (the hash functions, see below). However, the protocol from [35] seems more computationally intensive[1] for the quantum prover compared to the approach in [27]. For this reason, we only consider an experimental implementation of [27].

In this paper, we advance past the experimental work for the simpler learning with errors protocol in [22] to eliminate interaction and implement the protocol of [27] using 11 qubits on an ion-trap quantum computer. Our results are also complementary to the recent experimental work [23], which implements a simpler version [21] of Mahadev's interactive protocol for the classical verification of quantum computations [15] (but does not experimentally implement the required interaction). In each of our experiments, the quantum device's success rate in answering the verifier's challenges significantly exceeds that of the best possible classical strategy. This therefore verifies our device's nonclassical behavior and serves as a noninteractive proof of quantumness. We also comment on the possibility of scaling up this experiment to larger devices as a test of quantum mechanics, where we present this noninteractive protocol as a valuable alternative, which potentially has a much improved asymptotic scaling.

II. BACKGROUND ON CRYPTOGRAPHY

The noninteractive protocol of [27] relies on two cryptographic primitives: *trapdoor claw-free functions* (TCFs) and *hash functions* [38].

A TCF, denoted f, is a 2-to-1 function. In other words, there exist exactly two preimages x_0, x_1 that map to the same image $w = f(x_0) = f(x_1)$. The pair (x_0, x_1) is referred to as a *claw*. The "claw-free" property of a TCF is that, given the description of f (for instance, a circuit which evaluates f), it should be intractable to find a claw. In other words, no polynomial time classical (or quantum) algorithm can find a tuple (x_0, x_1, w) such that $f(x_0) = f(x_1) = w$. Finally, the *trapdoor* is a secret information that allows one to efficiently invert the function, recovering x_0 and x_1 from w.

The TCF we consider in this paper is based on the LWE problem [17,39]. In short, this problem is that of solving an *approximate* system of linear equations over the integers modulo q, denoted $\mathbb{Z}_q$. Explicitly, given an $m \times n$ matrix $A \in \mathbb{Z}_q^{m\times n}$ with entries modulo q and an m-dimensional vector $y = As + e \in \mathbb{Z}_q^m$ with entries modulo q, where e is a vector with small entries, known as the error vector, the problem is to solve for $s \in \{0, 1\}^n$. The entries in the error vector, e, are sampled from a discrete Gaussian distribution of small width (centered around zero). The LWE problem is conjectured to be intractable for both classical and quantum computers (i.e., it cannot be solved in polynomial time). This is known as the *LWE assumption* [39]. The assumed intractability forms the basis for defining a TCF. The specific TCF we consider

[1]Without going into details about how the protocol from [35] works, the reason it appears more computationally intensive is because the quantum prover in that protocol is required to perform a quantum Fourier transform (QFT) as well as coherently decode noisy codewords for a Reed-Solomon code. The circuits for these operations are comparable to the circuits for performing Shor's algorithm [36,37] and so would not be suitable for near-term devices. In contrast, as we explain later, the protocol from [27] does not require the use of the QFT and only requires the coherent evaluation of a class of functions that can be implemented with short depth circuits [20].

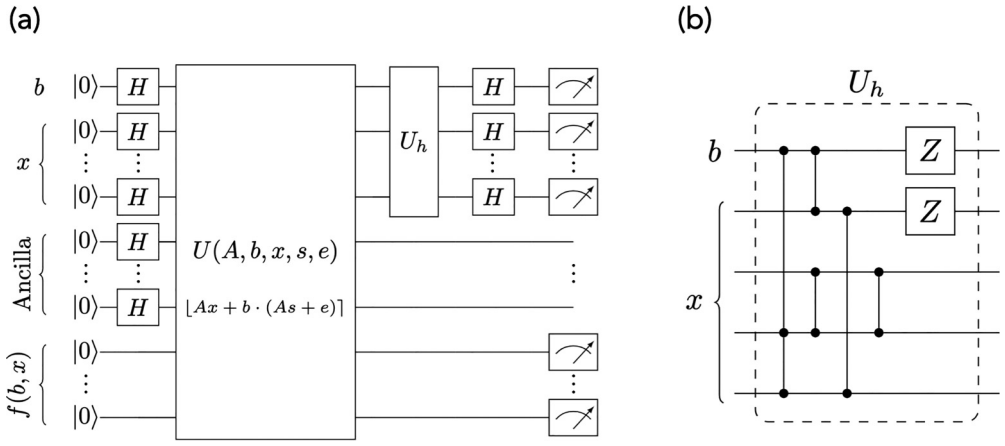

FIG. 1. Circuit diagrams for the prover's operations (a) and the hash function used (b) in the protocol. The prover first evaluates the TCF f and the hash function H on a superposition of all possible inputs. In (a), $U(A, b, x, s, e)$ denotes the operations used to evaluate the TCF in Eq. (1) and U_h denotes the operations used to evaluate the cryptographic hash function in Eq. (2), which is illustrated explicitly in (b). Details about the implementation of $U(A, b, x, s, e)$ can be found in the supplementary information of [22]. The prover then measures the registers of qubits storing b and x in the Hadamard basis and the register storing the result of evaluating the TCF in the standard basis.

here was also used in [20,22]. Starting from an LWE sample consisting of a matrix A and vector $y = As + e$, the function is defined as

$$f(b, x) = \lfloor Ax + by \rceil. \tag{1}$$

Here, $b \in \{0, 1\}$ is a single bit while $x \in \mathbb{Z}_q^n$ is a vector of dimension n with entries modulo q. Additionally, $\lfloor \cdot \rceil$ denotes a rounding operation, which can be understood as taking the most significant bits (MSBs) of the entry being rounded (for more details, see the related learning with rounding problem [40,41]). In this case, $\lfloor Ax + by \rceil$ corresponds to simply taking the most significant bit of each component of the vector $Ax + by$. Notice that here the claw is determined by $f(0, x_0) = f(1, x_1)$ where $x_1 = x_0 - s$.

The second type of cryptographic function we consider is the hash function. Hash functions are a fundamental tool in cryptographic protocols and are usually modeled as *random oracles*. An oracle function, $h : \{0, 1\}^* \to \{0, 1\}$, is a function for which one is not given an explicit description and instead queries it in a black box manner. Here $\{0, 1\}^*$ denotes bitstrings of arbitrary length. A random oracle refers to the fact that the oracle function is chosen uniformly at random from the set of all functions (or rather, for each input length n, one chooses a random function from $\{0, 1\}^n$ to $\{0, 1\}$). As it is often difficult to prove the security of a protocol with respect to a concrete instantiation of a hash function, one instead proves security in the *random oracle model* [42]. This simply means that the hash function is modeled as a random function, which all parties in the protocol can evaluate. Classically, this means querying with some input x and obtaining the output $h(x)$. In the quantum case, however, it is possible to query the random oracle in superposition [43]. In other words, when performing a quantum query, the state $\sum_x \alpha_x |x\rangle|y\rangle$ is mapped to $\sum_x \alpha_x |x\rangle|y \oplus h(x)\rangle$. Here, we restricted the output of the oracle to 1 bit, as this is the type of function used in the protocol of [27].

In the random oracle model and together with the LWE assumption, the protocol in [27] is a noninteractive test of quantum mechanics. When instantiating this protocol, we considered the TCF from Eq. (1) and a simple hash function represented as a low-degree polynomial. Ideally, one would use a hash function standardized by NIST, such as SHA-256 or SHA-3 [44,45]. However, those hash functions would require large numbers of qubits and gates in order to implement. For this reason, we propose using a small circuit, representing either a low-degree polynomial or a random (classical) circuit of short depth. This takes inspiration from the low-complexity hash functions introduced in [46] as well as the low-complexity one-way function of Goldreich [47]. Specifically, for our implementation we utilized the hash function

$$H(b, x) = b + x_1 + bx_1 + x_1x_4 + bx_3x_4, \tag{2}$$

where x_i denotes the ith bit of the binary representation of x. It should be noted that in our implementation, $x = x_1x_2x_3x_4$ is 4 bits long. A circuit diagram for this hash function, where the computation of the hash is performed in phase, is depicted in Fig. 1(b).

III. TEST OF QUANTUMNESS

A. Detailed protocol

With this background, we can now describe the protocol from [27] in more detail. A high-level circuit diagram depicting the prover's operations is displayed in Fig. 1. Recall that the protocol is noninteractive, in the sense that it only consists of one challenge message from the verifier to the prover, followed by the prover's response. Additionally, one assumes that the hash function is chosen before the start of the protocol and is known to both the verifier and the prover.

The protocol starts with the verifier generating an LWE instance (A, y) (together with a trapdoor [48,49]), that defines the TCF from Eq. (1) and sending the instance to the prover (while keeping the trapdoor secret). The prover is then required to evaluate the TCF f and the hash function H on a superposition of all possible inputs (consisting of the bit b and the string x). The TCF is evaluated in the computational basis, while the hash function is evaluated *in phase*. In other words,

the prover prepares the state

$$\sum_{b,x}(-1)^{H(b,x)}|b,x\rangle|f(b,x)\rangle, \tag{3}$$

suitably normalized. The prover then measures the register of qubits storing the result of evaluating the TCF, denoting the classical output as w, resulting in the state

$$\frac{1}{\sqrt{2}}(|0,x_0\rangle+(-1)^{H(0,x_0)+H(1,x_1)}|1,x_1\rangle)|w\rangle, \tag{4}$$

where $(0,x_0)$ and $(1,x_1)$ are the two preimages of $w=f(0,x_0)=f(1,x_1)$. Finally, the prover measures the qubits in the input registers storing b and x in the Hadamard basis. The prover's operations are depicted in Fig. 1(a). Denoting the first bit in the measurement outcome as z and the remaining bits as the string d, it can be shown that the following equation will be satisfied:

$$d\cdot(x_0\oplus x_1)=z\oplus H(0,x_0)\oplus H(1,x_1). \tag{5}$$

When the verifier sent the LWE instance to the prover, the challenge for the prover was to produce a tuple (w,z,d), such that Eq. (5) is satisfied. The quantum strategy of the prover, outlined here, does indeed produce such a tuple and this will be the prover's response to the verifier. The verifier uses the trapdoor to invert f on w, obtaining $(0,x_0)$ and $(1,x_1)$. With this, and the prover's response, the verifier checks Eq. (5), accepting if it is satisfied and rejecting otherwise. Note that while we presented the prover as performing its two measurements in sequence, the measurements can in fact be performed at the same time, as depicted in Fig. 1(a).

Let us provide some intuition for why a classical prover cannot succeed in the above protocol. The reason has to do with the intractability of finding a claw for the TCF and the fact that, classically, the random oracle (representing the hash function) can only be queried on a single input at a time (in contrast to the quantum case, where it is possible to query it on a superposition). We know that no efficient classical prover can produce a tuple (w,x_0,x_1), with $f(0,x_0)=f(1,x_1)=w$. Of course, in the protocol, the prover is merely required to produce a valid equation in the preimages of w, which can in principle be easier than finding a claw. However, in this case the use of the hash function precludes this possibility. A classical prover cannot compute both $H(0,x_0)$ and $H(1,x_1)$, as this would require querying the oracle on both points, meaning that the prover had obtained a claw. This then means that at least one of $H(0,x_0)$, $H(1,x_1)$ will be random and so a classical prover's probability of finding a valid equation will be $1/2$. By simply repeating the protocol multiple times, the classical prover's probability of succeeding in all challenges becomes negligible. The reason this argument fails for quantum provers is because quantum provers can query both the TCF and the random oracle in superposition. Indeed, this is precisely what is leveraged in the protocol in order to produce a valid equation. For the full proof of classical hardness, we refer the reader to [27].

B. Implementation of protocol

To concretely describe how the quantum prover executes the protocol, we now describe the quantum operations performed, which are illustrated at a high level in Fig. 1. Details about the implementation of $U(A,b,x,s,e)$ can be found in the supplementary information of [22]. We summarize this information here for completeness. The crucial portions of the algorithm are the evaluation of the TCF, represented by $U(A,b,x,s,e)$, and the evaluation of the hash function, represented by U_h. The latter is explicitly defined in Fig. 1(b), so we focus on the former. To compute the TCF, the prover utilizes four total registers of qubits: one for each of the b and x inputs, one for the output, and an ancilla register to assist in the computation, requiring a total of $N=1+n\log_2(q)+\log_2(q)+m$ qubits. Here, the bit b is stored in a single qubit while the n-dimensional vector x with entries modulo q requires $n\log_2(q)$ qubits. As we shortly describe, the ancilla register only stores one entry of an m-dimensional vector with entries modulo q at a time so that it uses $\log_2(q)$ qubits. Finally, the output register is an m-dimensional vector with binary entries; hence it requires m qubits. For our choices of parameters $n=2, m=4, q=4$, this results in $N=11$ qubits.

Recall that the prover wishes to compute $f(b,x)=\lfloor Ax+by\rfloor$ coherently, where this is an m-dimensional vector when A is an $m\times n$ matrix. The prover computes this vector one entry at a time, allowing it to reuse the ancilla qubits for each entry. More precisely, the ith component of this vector is the most significant bit of $\langle a_i,x\rangle+by_i$, where a_i denotes the ith row of the matrix A and $\langle\cdot,\cdot\rangle$ denotes the inner product modulo q. The prover can compute this in the ancilla register in the Fourier basis via several controlled rotation gates (controlled on the input qubits). Following this, the prover converts the ancilla register to the computational basis by applying the inverse quantum Fourier transform, "copies" the most significant bit from the ancilla register to the output register via another controlled rotation, and converts the ancilla register back into the Fourier basis. Now, the output register stores the ith component of the desired vector. To compute the remaining entries, the prover reverses this computation on the ancilla register and proceeds for the other rows of the matrix. This resetting of the ancilla allows the prover to reuse the qubits, significantly reducing the qubit usage required for the protocol. Precisely, without the ancilla, the number of qubits would scale as $1+n\log_2(q)+m\log_2(q)+m$, where $m>n$.

This also exhibits a tradeoff between the number of qubits and the depth of the quantum circuit. Namely, one can use more ancilla qubits to simultaneously compute multiple entries of the vector $f(b,x)$, thus decreasing the depth. This flexibility makes the protocol amenable to implementation on different devices, for which one of qubit count or depth may be more costly. Another potential tradeoff could be in the cost of reversing the computation on the ancilla register versus executing a mid-circuit reset of the ancilla qubits (similarly to the protocol in [18]). This could be useful for devices in which a mid-circuit reset is less costly than performing the gates required for reversing the computation coherently. We remark that this mid-circuit reset differs from the mid-circuit measurement approach for these protocols [22] because we are not required to perform gates on the postmeasurement ancilla qubits, potentially allowing for lower error rates [26].

We implement the quantum prover's circuits (Fig. 1) using an ion-trap quantum computer [22,50,51] to test our protocol. The quantum computer consists of 13 qubits made from a

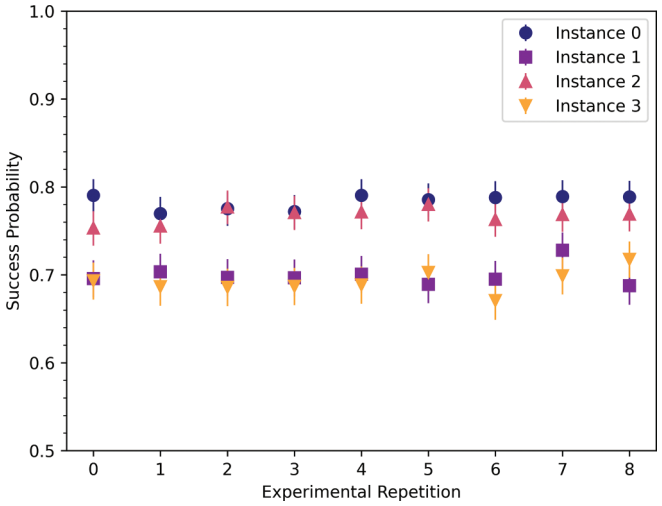

FIG. 2. Results of the protocol on four different LWE instances from Table I. The experiment is run nine times (experimental repetitions) for each instance with each repetition covering 2000 executions of the experiment. The data were collected 2000 executions at a time, which can indicate random fluctuations between each experimental repetition. The best possible success probability for a classical prover is 0.5 while it is 1.0 for an honest quantum prover. The error bars are computed using a binomial proportion 95% confidence interval via the Clopper-Pearson method as implemented in SCIPY.STATS.

linear chain of 15 $^{171}Yb^{+}$ ions that are laser cooled to near the motional ground state. In our protocol, we utilize 11 of these 13 available qubits. The system is capable of applying a universal gate set consisting of arbitrary single-qubit rotations on any target qubit as well as two-qubit Mølmer-Sørensen gates [52] on any arbitrary pair of qubits. The quantum circuits are implemented via the consecutive application of native single and two-qubit gates using individual optical addressing, where the fidelities of native single- and two-qubit gates are given by 99.98 and 98.5–99.3%, respectively [51]. Individual-qubit readout is performed with high fidelity at the end of circuit operations via state-dependent fluorescence detection [53].

IV. RESULTS

The results for implementing the protocols are displayed in Fig. 2. Here, we ran the experiment for several different choices of the matrix A and error vector e in the LWE instance, detailed in Table I using 11 qubits.

TABLE I. Details of the LWE instances. Note that the entries are transposed and for all instances we use $s^{\intercal} = (0 \quad 1)$.

Instance	$A^{\intercal}$	$e^{\intercal}$	$(As+e)^{\intercal}$
0	$\begin{pmatrix}0&2&0&1\\2&0&1&2\end{pmatrix}$	(0 1 0 0)	(0 3 0 1)
1	$\begin{pmatrix}0&2&3&2\\2&3&0&0\end{pmatrix}$	(0 0 0 1)	(0 2 3 3)
2	$\begin{pmatrix}2&0&0&1\\0&3&2&1\end{pmatrix}$	(1 0 1 0)	(3 0 1 1)
3	$\begin{pmatrix}0&1&3&0\\3&0&0&2\end{pmatrix}$	(0 0 0 1)	(0 1 3 1)

TABLE II. Gate counts for different LWE instances. The gate counts are given for the unoptimized (original) algorithm and the optimized algorithm with the optimizations detailed in the Appendix. We also note the number of multi-qubit operations in each of these cases.

	Instance			
	0	1	2	3
Unoptimized	73	75	81	79
Optimized	42	56	47	56
Multi-qubit unoptimized	45	47	53	51
Multi-qubit optimized	24	34	29	34

Furthermore, we repeat the experiment nine times, with each repetition covering 2000 executions of all four LWE instances in Table I. Hence, we obtain the success probabilities seen in Fig. 2. We note that instances 0 and 2 perform better due to optimizations reducing the gate count for the implementation of the TCF based on those instances. In particular, we achieved an approximately 42% decrease in gate count due to optimization for instances 0 and 2. In contrast, the same optimization reduced the gate count of instances 1 and 3 by only approximately 25 and 29%, respectively. The gate counts before and after optimization can be found in Table II. In general, the gate count scales as $O(mn\log_2 q)$. These optimizations were achieved by finding simpler ways to implement modular arithmetic operations for special cases for the form of the matrix A and vector $As+e$. This results in simplifying a number of controlled rotation gates into only one or two CNOT gates. The special cases we optimized for were determined after sampling A and $As+e$ and examining recurring patterns in their structure. It is expected that other choices of LWE instances could be optimized in a similar manner, although a separate analysis would likely be required. The optimizations are detailed in the Appendix.

V. DISCUSSION

We know from [27] that a classical adversary can succeed in the verifier's challenge with probability at most 0.5. Thus, we see that the results exceed this classical success probability for each LWE instance used. In particular, for each instance, we exceed this bound by between 15σ and 25σ, where σ denotes the standard deviation of the distribution of observed success probabilities under the null hypothesis that the prover is classical with success probability 0.5. Here, σ can be computed using the normal approximation to the binomial distribution as $\sigma = 1/(2\sqrt{N})$, where N is the number of executions of the experiment. The corresponding statistical significance with which the null hypothesis of a classical prover is rejected exceeds $p < 10^{-50}$, providing strong evidence of a quantum prover. Thus, our results significantly surpass the threshold for classical behavior, emphasizing our success in implementing a test of quantum mechanics experimentally, albeit on a small number of qubits. In particular, this confirms the quantum behavior of the device it was executed

on given the cryptographic security of our TCF and hash function.

Moreover, these results match reasonably well with estimates of the success probability based on known gate fidelities of the device. Namely, assuming a single-qubit gate fidelity of 99.98% and a two-qubit gate fidelity of 98.9% (the midpoint of the range 98.5–99.3%) [51], then a rough estimate based on the optimized gate count results in the following success probabilities: 76, 68, 72, and 68% for instances 0, 1, 2, and 3, respectively. Thus, these estimates agree roughly with the measured success probabilities.

We can also compare these results directly to those of [22]. Recall that for the "interference measurement" case considered in [22], a similar algorithm to the one in this paper is implemented. Namely, the circuit is almost the same as in Fig. 1(a) but with the hash function U_h removed. Moreover, the verifier checks the equation

$$d \cdot (x_0 \oplus x_1) = z,$$

as opposed to Eq. (5) in this paper. When implementing this "interference measurement" as a part of the interactive protocol with mid-circuit measurements, the success probability is approximately 71% on average [22]. In contrast, our noninteractive implementation has success probability of approximately 74% on average, with some instances reaching up to 77–78%. Repeating the experiment multiple times, this is a significant advantage. Thus, this indicates the benefit of using this noninteractive protocol over the interactive protocol via mid-circuit measurements.

VI. OUTLOOK

We implemented an efficient noninteractive test of quantumness with an ion-trap quantum computer and obtained results which exceed the threshold required for demonstrating nonclassical behavior under certain cryptographic assumptions. Since our implementation used 11 qubits, this does not constitute a certification of quantum advantage and is instead certifying quantum-mechanical behavior within the device. For a demonstration of quantum advantage, one would have to use a large enough instance of a claw-free function, for which classically "breaking" the underlying cryptographic task takes longer than the time it takes to run the experiment with a quantum device. The circuit complexity of implementing this function would be the dominating cost for the quantum prover's strategy, where the best achievable circuit sizes scale as $O(n \log n)$ [18]. Meanwhile, as we have shown with our implementation, the hash function can be implemented as some low-degree polynomial, or as a small random (classical) circuit. In particular, a quantum circuit with linear circuit complexity $O(n)$ will suffice in practice. Thus, the circuit complexity for both the interactive and noninteractive versions of the protocol is on the same order of magnitude. As such, the estimated numbers of qubits and circuit sizes for demonstrating quantum advantage with such a protocol are similar to those described in [18,22], namely $\approx 10^3$ qubits and $\approx 10^5$ layers of depth. As discussed previously, there is also a tradeoff between depth and qubit count, which one can leverage depending on which is more costly for the quantum computer used. The main advantage of our protocol and implementation compared to those protocols is the fact that interaction is not required for performing the test of quantumness, which reduces an additional potential barrier towards scaling these protocols up. Thus, this noninteractive protocol provides a promising alternative for implementing cryptographic proofs of quantumness in the future. Given this, as well as recent results aiming to reduce the costs of the quantum prover's implementation of the claw-free function [19,20], there are reasons to expect that these protocols could be used to certify quantum advantage on future generations of NISQ devices.

Finally, we note that the techniques used here have numerous further applications, not only in verifying quantum advantage once at scale, but also certifiable random number generation [16] and the classical verification of arbitrary quantum computation [15], for which noninteractive protocols have been achieved [27,54,55]. Although interactive protocols can also be utilized to accomplish these tasks and have been experimentally demonstrated [22,23], we emphasize that the noninteractive approach used here is simpler and thus more likely to be scalable. Thus by removing the additional barrier that interaction creates, these noninteractive protocols yield a promising path towards realizing tests of quantumness, randomness, and delegated computation on future quantum devices.

ACKNOWLEDGMENTS

The authors thank Gregory Kahanamoku-Meyer for valuable input. This work is supported by Air Force Office of Scientific Research Young Investigator Program (AFOSR YIP) Grant No. FA9550-16-1-0495, the Simons Foundation (TV Grant No. 828076), the National Science Foundation (NSF) Quantum Leap Challenge Institute (QLCI) program (Grant No. OMA-2016245), the Institute for Quantum Information and Matter (IQIM), an NSF Physics Frontiers Center (Grant No. PHY-1125565) with support of the Gordon and Betty Moore Foundation (Grant No. GBMF-12500028), Dr. Max Rössler, the Walter Haefner Foundation, the ETH Zürich Foundation, a Caltech Summer Undegraduate Research Fellowship, the Army Research Office (ARO) through the Intelligence Advanced Research Projects Activity (IARPA) LogiQ program, the National Science Foundation Software-Tailored Architecture for Quantum Co-design program, the U.S. Department of Energy Quantum Systems Accelerator program, the AFOSR MURI on Scalable Certification of Quantum Computing Devices and Networks (Grant No. FA9550-18-1-0161), the AFOSR MURI on Dissipation Engineering in Open Quantum Systems (Grant No. FA9550-19-1-0399), and the ARO MURI on Modular Quantum Circuits (Grant No. W911NF1610349).

APPENDIX: OPTIMIZATIONS

In this section, we detail the optimizations performed to decrease the gate count. After sampling the learning with errors instance, consisting of the matrix A and the vector $As + e$, these optimizations were made to simplify implementations of modular arithmetic operations for special cases for the form of A and $As + e$.

Specifically, one case we considered is when the ith row of A, denoted by a_i, is of the form $a_i = (0 \quad a)$ and the ith entry of $y = As + e$ is $y_i = 0$. As seen in Table I, this structure occurs several times for our sampled instances. We want to optimize the computation of the MSB of $\langle a_i, x\rangle + by_i$ for any input vector $x \in \mathbb{Z}_4^2$, where all operations are done modulo 4. Writing $x = (x_1 \quad x_2)^\intercal$, notice that, for our special case, this is simply the MSB of ax_2. It then remains to analyze all cases of $a, x_2 \in \mathbb{Z}_4$ to find a simpler way of implementing this operation. It is important to note that in our implementation, the vector x is stored in a quantum state in order to evaluate the trapdoor claw-free function in superposition while A (and thus the entry a) is stored classically. Thus, we can classically condition on each case of $a \in \mathbb{Z}_4$.

If $a = 0$, then the MSB of ax_2 is clearly zero. This requires no quantum operations to be performed. If $a = 1$, then $ax_2 = x_2$. The MSB of this is simply the MSB of x_2, which we can copy to the result register via a CNOT gate controlled on the qubit storing the MSB of x_2. The analysis becomes slightly more complicated for $a = 2, 3$. If $a = 2$, we notice that for x_2 equal to 0 or 2, then $2x_2 \bmod 4 = 0$ while for x_2 equal to 1 or 3, then $2x_2 \bmod 4 = 2$. Thus, the MSB of $2x_2$ is nonzero only for the cases of $x_2 = 1, 3$. However, notice that the least significant bit (LSB) of 1 and 3 is 1 while the LSB of 0 and 2 is zero. Thus, we can obtain the MSB of $2x_2$ by using a CNOT gate controlled on the qubit storing the LSB of x_2. Finally, if $a = 3$, we notice that

$$ax_2 \bmod 4 = 3 \times 0 \bmod 4 = 0, \tag{A1}$$

$$ax_2 \bmod 4 = 3 \times 1 \bmod 4 = 3, \tag{A2}$$

$$ax_2 \bmod 4 = 3 \times 2 \bmod 4 = 2, \tag{A3}$$

$$ax_2 \bmod 4 = 3 \times 3 \bmod 4 = 1. \tag{A4}$$

Notice that the MSB of $3x_2$ is nonzero only for the cases of x_2 equal to 1 or 2. Thus, we can obtain the MSB of $3x_2$ by using two CNOT gates: one controlled on the qubit storing the MSB of x_2 and another controlled on the qubit storing the LSB of x_2.

Here, we have simplified the evaluation of the TCF for this special case to only one or two CNOT gates. A similar analysis holds for the case when a_i is of the form $a_i = (a \quad 0)$ and $y_i = 0$, in which case we consider x_1 in the above analysis instead of x_2.

Another special case we considered was when a_i is of the form $a_i = (2 \quad 0)$ and $y_i = 3$. Again, we see in Table I that this case occurs several times. Similarly to before, write $x = (x_1 \quad x_2)^\intercal$. This time, the MSB of $\langle a_i, x\rangle + by_i$ is the MSB of $2x_1 + 3b$. It is important to note that b is stored in a quantum state in order to evaluate the TCF in superposition. We first analyze the cases of $b = 0$ and 1 separately. If $b = 0$, notice that the analysis is exactly the same as the base of $a = 2$ above. Namely, the MSB of $2x_1$ can be obtained by using a CNOT gate controlled on the qubit storing the LSB of x_1. On the other hand, if $b = 1$, then for x_1 equal to 0 or 2, $2x_1 + 3 \bmod 4 = 3$ while for x_1 equal to 1 or 3, $2x_1 + 3 \bmod 4 = 1$. Thus, the MSB of $2x_2 + 3$ is nonzero only for the cases of $x_1 = 0, 2$, which each have an LSB of zero. Since b is stored in a quantum state, we must distinguish the two cases of $b = 0, 1$ by a quantum operation. Thus, we must still include the CNOT gate controlled on the qubit storing the LSB of x_1 in this case. Then, in order to compute the correct MSB for the case of $b = 1$, we can also add a CNOT gate controlled on the qubit storing b. In this way, if $b = 0$, only the CNOT for the LSB of x_1 is executed. Meanwhile, if $b = 1$, the MSB of $2x_2 + 3$ is nonzero only for cases when x_1 has an LSB of zero. Thus, combining these CNOTs will give the desired result. Hence, we have again reduced the evaluation of the TCF to use only two CNOT gates.

The final special case we considered is similar to the above but for the case when a_i is of the form $a_i = (2 \quad 0)$ and $y_i = 1$. The analysis for $b = 0$ is exactly the same as above. If $b = 1$, then we want to compute $\langle a_i, x\rangle + by_i = 2x_1 + 1$. For x_1 equal to 1 or 3, then $2x_1 + 1 \bmod 4 = 3$, and for x_1 equal to 0 or 2, then $2x_1 + 1 \bmod 4 = 1$. Thus, the MSB of $2x_1 + 1$ is nonzero only for the cases of $x_1 = 1, 3$. This perfectly aligns with the case of $b = 0$, so we only require one CNOT controlled on the qubit storing the LSB of x_1.

Overall, these optimizations led to an approximately 42% decrease in gate count due to optimization for instances 0 and 2 and a decrease of approximately 25 and 29% for instances 1 and 3, respectively.

[1] D. Aharonov and U. Vazirani, arXiv:1206.3686.

[2] S. Goldwasser, S. Micali, and C. Rackoff, in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali* (Association for Computing Machinery, New York, NY, US, 2019), pp. 203–225.

[3] S. Goldwasser and M. Sipser, in *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, US, 1986), pp. 59–68.

[4] A. Shamir, J. ACM **39**, 869 (1992).

[5] J. Watrous, Theor. Comput. Sci. **292**, 575 (2003).

[6] R. Jain, Z. Ji, S. Upadhyay, and J. Watrous, Commun. ACM **53**, 102 (2010).

[7] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson, in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali* (Association for Computing Machinery, New York, NY, US, 2019), pp. 373–410.

[8] L. Babai, L. Fortnow, and C. Lund, Comput. Complex. **1**, 3 (1991).

[9] R. Cleve, P. Hoyer, B. Toner, and J. Watrous, in *Proceedings. 19th IEEE Annual Conference on Computational Complexity, 2004* (IEEE, New York, 2004), pp. 236–249.

[10] Z. Ji, A. Natarajan, T. Vidick, J. Wright, and H. Yuen, Commun. ACM **64**, 131 (2021).

[11] L. Babai, in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, US, 1985), pp. 421–429.

[12] L. Fortnow, J. Rompel, and M. Sipser, Theor. Comput. Sci. **134**, 545 (1994).

[13] O. Goldreich, S. Micali, and A. Wigderson, J. ACM **38**, 690 (1991).

[14] A. Gheorghiu, T. Kapourniotis, and E. Kashefi, Theory Comput. Syst. **63**, 715 (2019).

[15] U. Mahadev, in *Proceedings of the 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, New York, 2018), pp. 259–267.

[16] Z. Brakerski, P. Christiano, U. Mahadev, U. Vazirani, and T. Vidick, in *Proceedings of the 2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)* (IEEE, New York, 2018), pp. 320–331.

[17] O. Regev, J. ACM **56**, 1 (2009).

[18] G. D. Kahanamoku-Meyer, S. Choi, U. V. Vazirani, and N. Y. Yao, Nat. Phys. **18**, 918 (2022).

[19] S. Hirahara and F. L. Gall, 46th International Symposium on Mathematical Foundations of Computer Science (MFCS 2021), F. Bonchi and S. J. Puglisi (Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021), Vol. 202, pp. 59:1–59:15.

[20] Z. Liu and A. Gheorghiu, Quantum **6**, 807 (2022).

[21] J. Carrasco, A. Elben, C. Kokail, B. Kraus, and P. Zoller, PRX Quantum **2**, 010102 (2021).

[22] D. Zhu, G. D. Kahanamoku-Meyer, L. Lewis, C. Noel, O. Katz, B. Harraz, Q. Wang, A. Risinger, L. Feng, D. Biswas *et al.*, arXiv:2112.05156.

[23] R. Stricker, J. Carrasco, M. Ringbauer, L. Postler, M. Meth, C. Edmunds, P. Schindler, R. Blatt, P. Zoller, B. Kraus *et al.*, arXiv:2203.07395.

[24] V. Sivak, A. Eickbusch, B. Royer, S. Singh, I. Tsioutsios, S. Ganjam, A. Miano, B. Brock, A. Ding, L. Frunzio *et al.*, arXiv:2211.09116.

[25] M. Riebe, H. Häffner, C. Roos, W. Hänsel, J. Benhelm, G. Lancaster, T. Körber, C. Becher, F. Schmidt-Kaler, D. James *et al.*, Nature (London) **429**, 734 (2004).

[26] S. Moses, C. Baldwin, M. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn *et al.*, Phys. Rev. X **13**, 041052 (2023).

[27] Z. Brakerski, V. Koppula, U. Vazirani, and T. Vidick, arXiv:2005.04826.

[28] A. Fiat and A. Shamir, in *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques* (Springer, New York, 1986), pp. 186–194.

[29] F. Arute *et al.*, Nature (London) **574**, 505 (2019).

[30] H.-S. Zhong, H. Wang, Y.-H. Deng, M.-C. Chen, L.-C. Peng, Y.-H. Luo, J. Qin, D. Wu, X. Ding, Y. Hu *et al.*, Science **370**, 1460 (2020).

[31] S. Aaronson and L. Chen, arXiv:1612.05903.

[32] S. Aaronson and S. Gunn, arXiv:1910.12085.

[33] B. Barak, C.-N. Chou, and X. Gao, arXiv:2005.02421.

[34] X. Gao, M. Kalinowski, C.-N. Chou, M. D. Lukin, B. Barak, and S. Choi, arXiv:2112.01657.

[35] T. Yamakawa and M. Zhandry, arXiv:2204.02063.

[36] P. W. Shor, in *Proceedings 35th Annual Symposium on Foundations of Computer Science* (IEEE, New York, 1994), pp. 124–134.

[37] P. W. Shor, SIAM Rev. **41**, 303 (1999).

[38] S. Goldwasser, S. Micali, and R. L. Rivest, in *Advances in Cryptology: Proceedings of CRYPTO '84* (IEEE, Singer Island, FL, US, 1984), p. 467.

[39] O. Regev, *2010 IEEE 25th Annual Conference on Computational Complexity, Boston, Massachusetts* (IEEE, 2010), pp. 191–204.

[40] A. Banerjee, C. Peikert, and A. Rosen, in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, New York, 2012), pp. 719–737.

[41] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs, in *Proceedings of the Annual Cryptology Conference* (Springer, New York, 2013), pp. 57–74.

[42] M. Bellare and P. Rogaway, in *Proceedings of the First ACM Conference on Computer and Communications Security* (Association for Computing Machinery, New York, NY, US, 1993), pp. 62–73.

[43] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry, in *International Conference on the Theory and Application of Cryptology and Information Security* (Springer, New York, 2011), pp. 41–69.

[44] National Institute of Standards and Technology, Fed. Inf. Process. Stand. Publ. **180**, 36 (2015).

[45] M. J. Dworkin *et al.*, Federal Information Processing Standard (NIST FIPS) National Institute of Standards and Technology, Gaithersburg, MD (2015).

[46] B. Applebaum, N. Haramaty-Krasne, Y. Ishai, E. Kushilevitz, and V. Vaikuntananthan, in *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, edited by C. H. Papadimitriou, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 67 (Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2017), pp. 71–731.

[47] O. Goldreich, in *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation* (Springer, New York, 2011), pp. 76–87.

[48] D. Micciancio and C. Peikert, in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, New York, 2012), pp. 700–718.

[49] C. Gentry, C. Peikert, and V. Vaikuntanathan, in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, US, 2008), pp. 197–206.

[50] M. Cetina, L. N. Egan, C. A. Noel, M. L. Goldman, A. R. Risinger, D. Zhu, D. Biswas, and C. Monroe, arXiv:2007.06768.

[51] L. Egan, D. M. Debroy, C. Noel, A. Risinger, D. Zhu, D. Biswas, M. Newman, M. Li, K. R. Brown, M. Cetina *et al.*, Nature (London) **598**, 281 (2021).

[52] K. Mølmer and A. Sørensen, Phys. Rev. Lett. **82**, 1835 (1999).

[53] S. Olmschenk, K. C. Younge, D. L. Moehring, D. N. Matsukevich, P. Maunz, and C. Monroe, Phys. Rev. A **76**, 052314 (2007).

[54] N.-H. Chia, K.-M. Chung, and T. Yamakawa, in *Proceedings of the Theory of Cryptography Conference* (Springer, New York, 2020), pp. 181–206.

[55] G. Alagic, A. M. Childs, A. B. Grilo, and S.-H. Hung, in *Theory of Cryptography Conference* (Springer, New York, 2020), pp. 153–180.