On the Quantum versus Classical Learnability of Discrete Distributions

Ryan Sweke,[1] Jean-Pierre Seifert,[2] Dominik Hangleiter,[1] and Jens Eisert[1,3,4]

[1]*Dahlem Center for Complex Quantum Systems, Freie Universität Berlin, D-14195 Berlin, Germany*
[2]*Department of Electrical Engineering and Computer Science, TU Berlin, D-10587 Berlin, Germany*
[3]*Helmholtz Center Berlin, D-14109 Berlin, Germany*
[4]*Department of Mathematics and Computer Science, Freie Universität Berlin, D-14195 Berlin, Germany*

(Dated: July 30, 2020)

Here we study the comparative power of classical and quantum learners for generative modelling within the Probably Approximately Correct (PAC) framework. More specifically we consider the following task: Given samples from some unknown discrete probability distribution, output with high probability an efficient algorithm for generating new samples from a good approximation of the original distribution. Our primary result is the explicit construction of a class of discrete probability distributions which, under the decisional Diffie-Hellman assumption, is provably not efficiently PAC learnable by a classical generative modelling algorithm, but for which we construct an efficient quantum learner. This class of distributions therefore provides a concrete example of a generative modelling problem for which quantum learners exhibit a provable advantage over classical learning algorithms. In addition, we discuss techniques for proving classical generative modelling hardness results, as well as the relationship between the PAC learnability of Boolean functions and the PAC learnability of discrete probability distributions.

1. INTRODUCTION

Since its introduction, Valiant's model of "Probably Approximate Correct" (PAC) learning [1], along with a variety of natural extensions and modifications, has provided a fruitful framework for studying both the computational and statistical aspects of machine learning [2, 3]. Importantly, the PAC framework also provides a natural setting for the rigorous comparison of quantum and classical learning algorithms [4, 5]. In fact, while the recent availability of "noisy intermediate scale quantum" (NISQ) devices has spurred a huge interest in the potential of quantum enhanced learning algorithms [6–8], it is interesting to note that there is a rich history of quantum learning theory, beginning as early as 1995 with the seminal work of Bshouty and Jackson [4, 9]. Despite this rich history, the majority of previous work on quantum learning theory has focused on the classical versus quantum learnability of different classes of Boolean functions, which provides an abstraction of supervised learning [3].

In this work, we study the classical versus quantum PAC learnability of discrete probability distributions. More specifically, at an informal level we explore the following question, from the perspective of both classical and quantum learning algorithms: Given samples from some unknown probability distribution, output with high probability an efficient algorithm for generating new samples from a good approximation of the original distribution. We refer to this task as *generative modelling*. Note that one could also consider the related problem not of generating new samples from a distribution, but of learning a description of the distribution itself – a problem known as *density modelling* [10–12].

Here, we focus exclusively on generative modelling for a variety of reasons. Firstly, from a purely classical perspective, modern heuristic models and algorithms for generative modelling, such as Generative Adversarial Networks (GANS) [13], Variational Auto-Encoders [14] and Normalizing Flows [15] have proven extremely successful, with a wide variety of practical applications, and as such understanding whether quantum algorithms may be able to offer an advantage for this task is of natural interest. Additionally, a variety of quantum models and algorithms for generative modelling have recently been proposed, such as Born Machines [16, 17], Quantum GANS [18–21] and Quantum Hamiltonian-based models [22], but remain ill-understood from a theoretical perspective. Furthermore, we know that there exist probability distributions which cannot be efficiently sampled from classically, but which can be efficiently sampled from using quantum devices [23–25]. In light of this fact, and the emergence of quantum algorithms for generative modelling, Ref. [17] has formalized the question, within the PAC framework, of whether there also exist classes of probability distributions which can be efficiently *learned* (in a generative sense) with quantum resources, but not with purely classical approaches. Our primary contribution in this work is to answer this question in the affirmative, through the explicit construction of a concept class of discrete probability distributions which, under the Decisional Diffie-Hellman (DDH) assumption [26], is provably not efficiently learnable from classical samples by a classical learning algorithm, but for which we provide an efficient quantum learning algorithm. This class of distributions therefore provides a concrete example of a generative modelling problem for which quantum learners exhibit a provable advantage over classical learning algorithms.

The following important points regarding the setting of the result are worth clarifying. Firstly, although it might seem natural to consider the learnability of probability distributions describing the outcome of quantum processes, such as the measurement of a parameterized quantum state or random quantum circuit, we focus exclusively in this work on probability distributions describing the outcomes of *classical* circuits. Apart from allowing us to exploit existing results concerning the hardness of learnability for specific classes of discrete probability distributions [27], this restriction also allows us to demonstrate a quantum generative modelling advantage for purely classical problems. Additionally, in the context of Boolean function learning, it is often of interest

to consider quantum learning algorithms which have access to *quantum examples* – in essence a superposition of input/output pairs from the unknown function to be learned [4]. In the setting we are concerned with here, it is also possible to consider a notion of *quantum samples* from a distribution, however once again we choose to restrict ourselves to quantum learning algorithms with access to classical samples from the unknown probability distribution, which provides the fairest comparison of quantum versus classical learners for generative modelling. Finally, it is important to stress that the efficient quantum learning algorithm which we provide is expected to require a universal fault-tolerant quantum computer, as it makes use of the exact efficient quantum algorithm for discrete logarithms [28]. As a result, we *do not* expect that the separation we show in this work can be experimentally demonstrated on NISQ devices. Studying the learnability of probability distributions generated by quantum processes, the power of learners with quantum samples, and the power of near-term quantum learning algorithms remain interesting open problems, and as such we will also discuss the consequences of our results and techniques for approaching these questions.

To provide a generative modelling task for which there exists a definitive provable separation between the power of quantum and classical learners, we rely heavily on techniques at the rich interface of computational learning theory and cryptography [2]. More specifically, we start from the prior work of Kearns et al. [27], who have shown that given any pseudorandom function collection it is possible to construct a class of probability distributions for which no efficient classical generative modelling algorithm exists. We show that in order for such a class of distributions to be efficiently quantum learnable, one requires a pseudorandom function collection for which there exists a quantum adversary, who in addition to distinguishing keyed instances of the pseudorandom function collection from random functions via membership queries, can also learn the secret key using only random examples. By using the DDH assumption as a primitive, we are then able to construct such a pseudorandom function collection via a slight modification of the Goldreich-Goldwasser-Micali (GGM) construction [29].

Although the classical hardness result of Kearns et al. [27] is a sufficient starting point for our purposes, we also address in this work the possibility of obtaining similar classical hardness results for generative modelling from primitives other than pseudorandom function collections. More specifically, we formulate and discuss conjectures concerning the possibility of proving classical hardness results for generative modelling from both *weak* pseudorandom function collections, and from existing hardness results for the PAC learnability of *Boolean functions*. Apart from being of conceptual interest, in the former case these considerations are motivated by the possibility of using such results to address questions concerning generative modelling with near-term quantum learners, as well as quantum learners with quantum samples. In the latter case, these considerations are motivated by a desire to understand better the relationship between the PAC learnability of discrete probability distributions, and the PAC learnability of Boolean functions.

From the above outline one can see that both the results and techniques of this work lie at the intersection of quantum machine learning, computational learning theory and cryptography. In particular, while our primary result is very much in the spirit of computational learning theory, and contributes new ideas and techniques in this vein, it is also certainly of interest to the quantum machine learning community, and largely motivated by a desire to understand more clearly the potential and limitations of quantum enhanced machine learning. As a result, in order for this work to be accessible to readers with differing backgrounds and interests, we will provide a detailed and pedagogical presentation of the foundational material necessary for understanding both the context and details of our main result.

We proceed in this work as follows: Firstly, we begin in Section 2 with an introduction to the PAC framework, both for concept classes consisting of Boolean functions, and for the generative modelling of concept classes consisting of probability distributions over discrete domains. Given these foundations, we conclude Section 2 with the statement of Question 1, which provides a precise technical description of the primary question that we address in this work, and which we have described informally above. With this in hand, we then proceed in Section 3 to answer Question 1 in the affirmative. More specifically, after providing an overview of the necessary cryptographic notions in Section 3.1, we present in Section 3.2 a technique due to Kearns et al. [27] for constructing from any pseudorandom function collection a distribution concept class which is provably not efficiently learnable by classical learning algorithms. This technique then allows us to construct in Section 3.3 a distribution concept class which, under the DDH assumption, is provably not efficiently learnable by any classical learning algorithm, but for which we provide explicitly an efficient quantum learner for the generative modelling task. Having fully addressed Question 1 at this point, we then shift gears and explore in Section 4 the possibility of obtaining classical generative modelling hardness results from primitives other than pseudorandom function collections. In particular, in Section 4.1 we discuss whether weak pseudorandom function collections would be sufficient, and in Section 4.2 we examine the relationship between PAC learnability of Boolean functions, and the PAC generative modelling of associated probability distributions. Finally, in Section 5 we summarize our results, and provide an overview of interesting related and open questions, focusing specifically on the setting of probability distributions generated by quantum processes.

2. QUANTUM AND CLASSICAL PAC LEARNING

In this section, we begin by defining the notion of *probably approximately correct* (PAC) learnability, both for concept classes consisting of Boolean functions, and concept classes consisting of probability distributions over discrete domains. As we will see, these notions provide a meaningful abstract framework for studying computational aspects of both supervised learning and

probabilistic/generative modelling respectively. While the main result of this work is concerned with the latter setting, we begin with the more familiar context of Boolean functions in order to introduce both the fundamental ideas, and a variety of oracle models which will be important throughout this work. Additionally, as mentioned in the introduction, after presentation of our main distribution learning results in Section 3, we will in Section 4.2 discuss in detail the relationship between PAC learnability of Boolean function classes, and PAC learnability of discrete distribution classes.

2.1. PAC Learning of Boolean Functions

Let us denote by $\mathcal{F}_n$ the set of all Boolean functions on n bits – i.e. $\mathcal{F}_n = \{f | f : \{0,1\}^n \to \{0,1\}\}$. Notice that any function in $\mathcal{F}_n$ can be specified via its truth table, and therefore $\mathcal{F}_n \simeq \{0,1\}^{2^n}$. We call any subset $\mathcal{C} \subseteq \mathcal{F}_n$ a *concept class*. For any $f \in \mathcal{F}_n$ we can define various types of classical and quantum oracle access to f. Classically, we define the *membership query* oracle $\mathrm{MQ}(f)$ as the oracle which on input x returns $f(x)$, and the *random example* oracle $\mathrm{PEX}(f, D)$ as the oracle which when invoked returns a tuple $(x, f(x))$, where x is drawn from the distribution D over $\{0,1\}^n$. It will also be useful to us later to define the oracle $\mathrm{RPEX}(f, D)$ which when invoked returns only $f(x)$, with x drawn from D. This can be summarized as follows:

$$\mathsf{Query}[\mathrm{MQ}(f)](x) = f(x), \tag{1}$$

$$\mathsf{Query}[\mathrm{PEX}(f, D)] = (x, f(x)) \text{ with } x \leftarrow D, \tag{2}$$

$$\mathsf{Query}[\mathrm{RPEX}(f, D)] = f(x) \text{ with } x \leftarrow D, \tag{3}$$

where we have used the notation $x \leftarrow D$ to indicate that x is drawn from D. Additionally, we define the *quantum membership query* oracle $\mathrm{QMQ}(f)$ as the oracle which on input $|x\rangle \otimes |y\rangle$ returns $|x\rangle \otimes |f(x) \oplus y\rangle$, and the *quantum random example* oracle $\mathrm{QPEX}(f, D)$ as the oracle which when invoked returns the quantum state $\sum_x \sqrt{D(x)}|f(x)\rangle$, where again D is some distribution over $\{0,1\}^n$. This can be summarized as

$$\mathsf{Query}[\mathrm{QMQ}(f)](|x\rangle \otimes |y\rangle) = |x\rangle| \otimes |f(x) \oplus y\rangle, \tag{4}$$

$$\mathsf{Query}[\mathrm{QPEX}(f, D)] = \sum_{x \in \{0,1\}^n} \sqrt{D(x)}|f(x)\rangle. \tag{5}$$

As it will be convenient later, we also define $\mathrm{MQ}(f, D) := \mathrm{MQ}(f)$ and $\mathrm{QMQ}(f, D) := \mathrm{QMQ}(f)$ for all distributions D. For a more detailed discussion of these oracles, and in particular the motivation behind their definitions and the relationships between them, we refer to Ref. [4]. Given these notions, we can then formulate the following definition of a PAC learner for a given concept class:

Definition 1 (PAC Learners)**.** *An algorithm $\mathcal{A}$ is an (ϵ, δ, O, D) PAC learner for a concept class $\mathcal{C} \subseteq \mathcal{F}_n$, if for all $c \in \mathcal{C}$, when given access to oracle $O(c, D)$, with probability at least $1 - \delta$, the learner $\mathcal{A}$ outputs a hypothesis $h \in \mathcal{F}_n$ such that*

$$\Pr_{x \leftarrow D}[h(x) \neq c(x)] \leq \epsilon. \tag{6}$$

An algorithm $\mathcal{A}$ is an (ϵ, δ, O) PAC learner for a concept class $\mathcal{C}$, if it is an (ϵ, δ, O, D) PAC learner for all distributions D.

Before continuing, it is useful to make some comments concerning this definition. Firstly, note that the above formulation allows us to consider both classical learning algorithms, with either classical membership query or classical random example oracle access, as well as quantum learning algorithms, with either classical or quantum oracle access of any type. Additionally, it is important to point out that for a given model of oracle access O we can consider either distribution dependent learners – i.e. learners which are required to succeed (in the sense of being probably approximately correct) only with respect to samples drawn from some fixed distribution D, or distribution independent learners, which should succeed with respect to samples drawn from all possible distributions. In light of these observations, we see that Definition 1 provides for us a flexible abstraction of supervised learning, which allows for the comparison of a variety of different learning algorithms, each of which models the supervised learning problem in a different context. In order to perform a meaningful *computational* comparison of these different learning algorithms, we need the following notions of sample and time complexity.

Definition 2 (Complexity of PAC Learners)**.** *The sample (time) complexity of an (ϵ, δ, O, D) PAC learner $\mathcal{A}$ for a concept class $\mathcal{C}$ is the maximum number of queries made by $\mathcal{A}$ to the oracle $O(c, D)$ (maximum run-time required by $\mathcal{A}$) over all $c \in \mathcal{C}$, and over all internal randomness of the learner. The sample (time) complexity of an (ϵ, δ, O) PAC learner $\mathcal{A}$ for a concept class $\mathcal{C}$ is the maximum number of queries made by $\mathcal{A}$ to the oracle $O(c, D)$ (maximum run-time required by $\mathcal{A}$) over all $c \in \mathcal{C}$, all possible distributions D and all internal randomness of the learner. Both an (ϵ, δ, O, D) and an (ϵ, δ, O) PAC learner for a concept class $\mathcal{C} \subseteq \mathcal{F}_n$ is called efficient if its time complexity is $\mathcal{O}(\mathrm{poly}(n, 1/\delta, 1/\epsilon))$.*

Given this, the following definition formalizes a variety of notions for the efficient PAC learnability of a concept class:

Definition 3 (Efficient PAC Learnability of a Concept Class)**.** *We say that a concept class $\mathcal{C}$ is efficiently classically (quantum) PAC learnable with respect to distribution D and oracle O if for all $0 < \epsilon, \delta < 1$ there exists an efficient classical (quantum) (ϵ, δ, O, D) PAC learner for $\mathcal{C}$. Similarly, $\mathcal{C}$ is efficiently classically (quantum) PAC learnable with respect to oracle O if for all $0 < \epsilon, \delta < 1$ there exists an efficient classical (quantum) (ϵ, δ, O) PAC learner for $\mathcal{C}$.*

For a complete overview of known results and open questions concerning classical versus quantum learnability of Boolean function concept classes, we again refer to Ref. [4].

2.2. PAC Learning of Discrete Distributions

In the previous section we provided definitions for the PAC learnability of concept classes consisting of Boolean functions, which provides an abstract framework for studying and comparing computational properties of different supervised learning algorithms. In this section, we formulate a generalization to concept classes consisting of discrete distributions, which builds on and refines the prior work of Refs. [17, 27], and provides an abstract framework for studying probabilistic modelling from a computational perspective. Additionally, this formulation allows us to state precisely the primary question that we address in this work. For simplicity (and without loss of generality) we will consider distributions over bit strings, and as such we denote the set of all distributions over $\{0,1\}^n$ as $\mathcal{D}_n$, and we call any $\mathcal{C} \subseteq \mathcal{D}_n$ a *distribution concept class*. We also denote the uniform distribution over $\{0,1\}^n$ as U_n. In order to provide a meaningful generalization of PAC learning to this setting, the first thing that we require is a meaningful notion of a query to a distribution. To do this, given some distribution $D \in \mathcal{D}_n$, we define the *sample* oracle $\mathrm{SAMPLE}(D)$ as the oracle which when invoked returns some x drawn from D. More specifically, we have that

$$\mathsf{Query}[\mathrm{SAMPLE}(D)] = x \leftarrow D. \tag{7}$$

Additionally, it is natural to define the *quantum sample* oracle $\mathrm{QSAMPLE}(D)$ via

$$\mathsf{Query}[\mathrm{QSAMPLE}(D)] = \sum_{x \in \{0,1\}^n} \sqrt{D(x)}|x\rangle. \tag{8}$$

In particular, note that given access to $\mathrm{QSAMPLE}(D)$ one can straightforwardly simulate access to $\mathrm{SAMPLE}(D)$ by simply querying $\mathrm{QSAMPLE}(D)$, and then performing a measurement in the computational basis. At this point, it is important to point out that unlike in the case of function concept classes – where what it means to "learn a function" is relatively straightforward – there are two distinct notions of what it means to "learn a distribution" [27]. Informally, given some unknown distribution D, as well as access to either a classical or quantum sample oracle, we could ask that a learning algorithm outputs an *evaluator* for D – i.e. some function $\tilde{D} : \{0,1\}^n \to [0,1]$ which on input $x \in \{0,1\}^n$ outputs an estimate for $D(x)$, and therefore provides an approximate description of the distribution. This is perhaps the most intuitive notion of what it means to learn a probability distribution, and one can indeed construct a corresponding notion of PAC learnability [27], for which a variety of results are known for different distribution concept classes [10–12, 27]. However, in many practical settings one might not be interested in learning a full description of the probability distribution (an evaluator for the probability of events) but rather in being able to generate samples from the distribution. As such, instead of asking for a description of the unknown probability distribution (an evaluator) we could ask that the learning algorithm outputs a *generator* for D – i.e. a probabilistic (quantum or classical) algorithm which when run generates samples from D. From a heuristic perspective we note that many of the most widely utilized probabilistic modelling architectures and algorithms, such as generative adversarial networks, are precisely learning algorithms of this type. Additionally, there has recently been a surge of interest in *quantum* learning algorithms of this type – so called *Born machines* [16, 17] – which are based on the simple observation that one can sample from a given distribution by preparing and measuring an appropriate quantum state (such as the state provided by the QSAMPLE oracle). While the learning of evaluators is certainly both interesting and important, with many open questions [12], in this work we will focus exclusively on the problem of learning generators for distribution concept classes. To this end, we start with the following definition, adapted from Refs. [17, 27], which formalizes the notion of an efficient generator:

Definition 4 (Efficient Generator)**.** *Given some probability distribution D over $\{0,1\}^n$, we say that a classical (quantum) algorithm GEN_D is an efficient generator for D if GEN_D produces samples in $\{0,1\}^n$ according to D, using $\mathrm{poly}(n)$ resources. In the case of a classical algorithm, we allow the generator to receive as input $m = \mathrm{poly}(n)$ uniformly random input bits.*

We say that a distribution concept class $\mathcal{C}$ can be efficiently classically (quantum) generated if for all $D \in \mathcal{C}$ there exists an efficient classical (quantum) generator for D. Additionally, again following Refs. [17, 27], we can define the following notion of an approximate generator, which is necessary for a meaningful notion of PAC learnability:

Definition 5 ((d, ϵ) Generator)**.** *Let d be some distance measure on the space of probability distributions over $\{0,1\}^n$, and D some probability distribution over $\{0,1\}^n$. Given some other distribution D' over $\{0,1\}^n$, as well as an efficient generator $\mathsf{GEN}_{D'}$ for D', we say that $\mathsf{GEN}_{D'}$ is an efficient (d, ϵ) generator for D if $d(D, D') \leq \epsilon$.*

In this work we will use primarily the Kullback-Leibler (KL) divergence, defined via[1]

$$d_{\mathrm{KL}}(D, D') := \sum_x D(x) \log \left(\frac{D(x)}{D'(x)} \right), \tag{9}$$

as well as the total variation (TV) distance

$$d_{\mathrm{TV}}(D, D') := \frac{1}{2} \sum_x |D(x) - D'(x)|. \tag{10}$$

We note that by virtue of its asymmetry the KL-divergence is not strictly a metric, however, via Pinsker's inequality we have that

$$d_{\mathrm{TV}}(D, D') \leq \ln(2) \sqrt{d_{\mathrm{KL}}(D, D')}. \tag{11}$$

For more details on the interpretation of these and other relevant distance measures, we refer to Ref. [30]. Given these preliminaries the following definition provides a natural generalisation of Definition 1 to the generative modelling context in which we are interested:

Definition 6 (PAC Generator Learners)**.** *A learning algorithm $\mathcal{A}$ is an (ϵ, δ, O, d) PAC generator learner for a distribution concept class $\mathcal{C}$, if for all $D \in \mathcal{C}$, when given access to oracle $O(D)$, with probability $1 - \delta$ the learner $\mathcal{A}$ outputs a (d, ϵ) generator $\mathsf{GEN}_{D'}$ for D.*

Once again, as illustrated in Fig. 1, it should be clear from Definition 6 that we can consider both classical and quantum learning algorithms, where in the quantum case the learner could have access to either the classical or quantum sample oracle. Additionally, as will be discussed in more detail below, note that we have not specified the type of generator – classical or quantum – which should be output by the learner. While perhaps counter-intuitive, it is possible to consider quantum learners which output classical generators, as well as classical learners which output descriptions of quantum generators. Now, given the above definition, we can define the sample/time complexity of a PAC generator learner, as well as the efficient PAC learnability of a distribution concept class, analogously to how we have defined these notions in Definitions 2 and 3.

Definition 7 (Complexity of PAC Generator Learners)**.** *The sample (time) complexity of an (ϵ, δ, O, d) PAC generator learner $\mathcal{A}$ for a distribution concept class $\mathcal{C}$ is the maximum number of queries made by $\mathcal{A}$ to the oracle $O(D)$ (maximum run-time required by $\mathcal{A}$) over all $D \in \mathcal{C}$, and over all internal randomness of the learner. An (ϵ, δ, O, d) PAC generator learner for a concept class $\mathcal{C} \subseteq \mathcal{F}_n$ is called efficient if its time complexity is $\mathcal{O}(\mathrm{poly}(n, 1/\delta, 1/\epsilon))$.*

Definition 8 (Efficient PAC Generator-Learnability of a Distribution Concept Class)**.** *We say that a distribution concept class $\mathcal{C}$ is efficiently classically (quantum) PAC generator-learnable with respect to oracle O and distance measure d if for all $\epsilon > 0$, $0 < \delta < 1$ there exists an efficient classical (quantum) (ϵ, δ, O, d) PAC generator learner for $\mathcal{C}$.*

For convenience, if the distance measure d and the oracle O is clear from the context, we will sometimes simply refer to an (ϵ, δ, O, d) PAC generator learner as a learner, and to a distribution concept class which is efficiently classically (quantum) PAC-generator learnable with respect to O and d as simply classically (quantum) learnable. Similarly, if a distribution concept class is provably *not* efficiently classically (quantum) PAC generator-learnable (with respect to some oracle and some distance measure) we will say that it is classically (quantum) hard to learn. Given these definitions, we are finally in a position to state precisely the primary question that we explore in this work:

Question 1. *Does there exist a distribution concept class $\mathcal{C}$, which can be efficiently classically generated, and which is efficiently quantum PAC generator-learnable with respect to the* SAMPLE *oracle and the KL-divergence, but which is not efficiently classically PAC generator-learnable with respect to the* SAMPLE *oracle and the KL-divergence?*

In the following section we will show, via an explicit construction of such a distribution concept class, that up to a standard cryptographic assumption, the answer to this question is "Yes". Before continuing though, there are a variety of important and relatively subtle observations to be made. Firstly, we note that Ref. [17] has defined "quantum learning supremacy" as the existence of a distribution concept class for which, for some distance measure d (such as the total variation distance), for all

[1] All logarithms in this work are base 2.

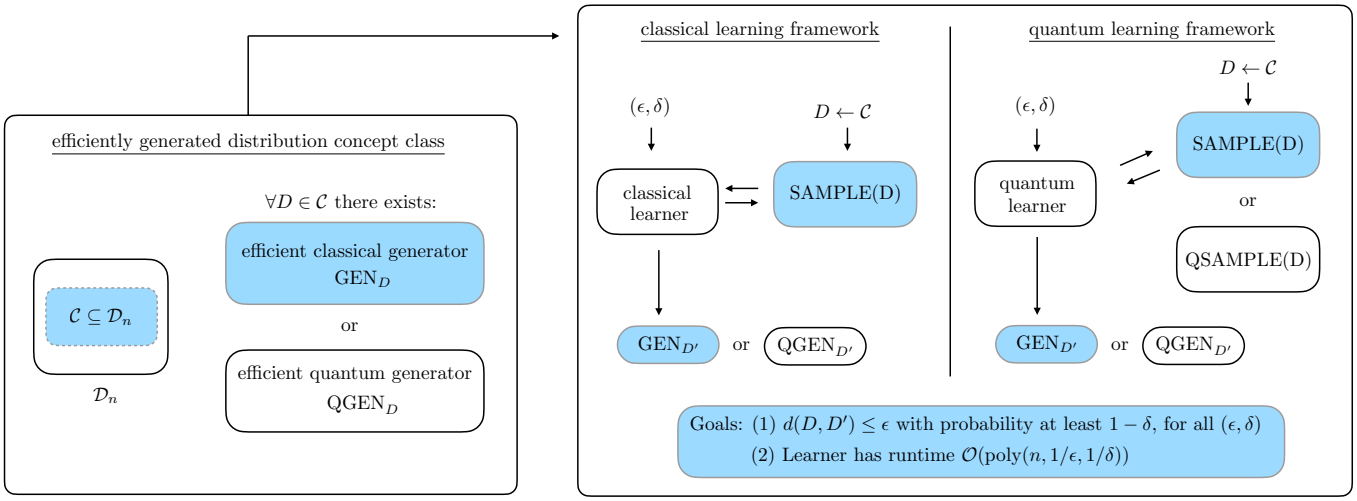

FIG. 1. An overview of the PAC framework for generative modelling. As discussed in the main text, one could consider either classically generated or quantum generated distribution concept classes, quantum learners with access to either classical or quantum sample oracles, and learners which output either classical or quantum hypothesis generators. In this work, we focus on the setting indicated by the shaded boxes – i.e. concept classes which are efficiently classically generated, quantum learners with access only to classical samples, and both classical and quantum learners which output classical hypothesis generators.

$0 < \delta < 1$, and for some *fixed* $\epsilon > 0$, there exists an efficient $(\epsilon, \delta, \mathrm{SAMPLE}, d)$ quantum PAC-generator learner, while for at least one value of $0 < \delta < 1$ there does not exist an efficient classical $(\epsilon, \delta, \mathrm{SAMPLE}, d)$ PAC generator learner. As such we see that a positive answer to Question 1 provides not only a clear example of what Ref. [17] has called "quantum learning supremacy", but in fact something slightly stronger, as a result of the fact that in order for a distribution concept class to be efficiently quantum PAC generator-learnable (as per Definition 8) there should exist an efficient $(\epsilon, \delta, \mathrm{SAMPLE}, d)$ quantum PAC-generator learner not just for some fixed $\epsilon > 0$, but for *all* $\epsilon > 0$.

Additionally, as illustrated in Fig. 1, it is also important to note that, unlike in Ref. [17], we have explicitly added the additional restriction that the distribution concept class be *efficiently classically generated* – i.e. that for all $D \in \mathcal{C}$ there exists an efficient *classical* generator GEN_D. If one's goal is simply to construct a distribution concept class which is efficiently quantum learnable, but not efficiently classically learnable, then in principle this restriction is not necessary, and one could indeed also consider concept classes which are efficiently quantum generated. However, we have chosen to consider this additional constraint here in order to make clear the conceptual distinction between generative modelling problems defined by underlying classical processes, and generative modelling problems defined by underlying quantum processes. This distinction is helpful in that it allows us to distinguish more clearly the type of problems for which we might expect quantum learners to demonstrate an advantage, and the type of problems for which we can possibly rule out such an advantage. In fact, as discussed in Ref. [17], we know that there exist distribution classes which can be efficiently quantum generated, but cannot be efficiently classically generated [23–25], and as such if one aims to demonstrate the existence of a distribution concept class which can be efficiently quantum learned, but cannot be efficiently classically learned, then one might think that such a distribution class would be a natural candidate. This may indeed be true, and in this work we have chosen to focus only on distribution concept classes which can be efficiently classically generated both to rule out the possibility of a quantum advantage due to the computational difficulty of sampling classically from the distributions in the concept class, and to demonstrate clearly that quantum learning algorithms can obtain a clear advantage even for problems defined by underlying classical processes. That said, as mentioned before it is important to note that even if a distribution class cannot be efficiently classically generated, this does not immediately rule out the existence efficient classical learners which are able to output descriptions of the efficient quantum generators. As such, despite the focus here on distribution classes which can be efficiently classically generated, the question of whether one can prove a quantum learning advantage for distribution classes which are specified by efficient quantum generators remains an interesting open question, which we discuss in Section 5.

Finally, also as illustrated in Fig. 1, we note that in Question 1 we have restricted both the classical and quantum learning algorithms to classical SAMPLE oracle access to the unknown distributions. Once again, as mentioned before, this is not strictly necessary, and one could also consider quantum learners with access to the QSAMPLE oracle. However, we have chosen to restrict ourselves here to classical SAMPLE oracle access both because this is the most natural abstraction of a typical applied generative modelling problem, and because this provides the "fairest" playing field on which to compare the power of classical and quantum learners. That said, understanding the additional power that quantum samples might offer a quantum generator learner is indeed also an interesting open question, which we formulate and discuss in Section 5.

3. A QUANTUM/CLASSICAL DISTRIBUTION LEARNING SEPARATION

In this section we answer Question 1 in the affirmative, providing the main result of this work. To this end, we use the result of Kearns et al. [27] as a starting point, who have shown that any pseudorandom function (PRF) can be used to construct a distribution concept class which is not efficiently classically PAC generator-learnable, with respect to SAMPLE and the KL-divergence. In addition, each distribution in the concept class defined by Kearns et al. admits an efficient classical generator, which is fully specified by a key of the underlying PRF. Given this insight, we design a keyed function which, under the decisional Diffie-Hellman (DDH) assumption for the group family of quadratic residues [26], is pseudorandom from the perspective of classical adversaries, but not pseudorandom from the perspective of quantum adversaries, who in addition to distinguishing keyed instances of the function from random with membership queries, can also learn the secret key using only random examples. Instantiating a slight modification of the Kearns construction with this DDH based PRF yields a distribution concept class which answers Question 1 in the affirmative, under the DDH assumption for quadratic residues.

We proceed by introducing all the necessary cryptographic primitives in Section 3.1. Equipped with these preliminaries, we then present in Section 3.2 the classical hardness result of Kearns [27], along with some important corollaries and modifications. Finally, given this result, in Section 3.3 we use the DDH assumption to explicitly construct a distribution concept class which, due to the results in Section 3.2, is provably not efficiently classically learnable, but for which we are able to construct explicitly an efficient quantum learner.

3.1. Cryptographic Primitives

We begin here with a brief overview of the cryptographic notions which are necessary to understand the constructions in the following sections. For a more detailed introduction to these concepts and constructions, we refer to Ref. [31]. The first notion that we need is an indexed collection of functions.

Definition 9 (Indexed Collection of Efficiently Computable Functions)**.** *Given some infinite set of parameterizations $\mathcal{P}$, we say that the set of functions $\{F_P \mid P \in \mathcal{P}\}$ is an indexed collection of efficiently computable functions if for all $P \in \mathcal{P}$ we have that $F_P : \mathcal{D}_P \to \mathcal{D}'_P$, and there exists:*

1. *An efficient (possibly probabilistic) instance generation algorithm $\mathcal{IG}$, which on input 1^n outputs some $P \in \mathcal{P}$, as well as a description of the the domain $\mathcal{D}_P$ and the codomain $\mathcal{D}'_P$.*
2. *An efficient evaluation algorithm which, for all $P \in \mathcal{P}$ and all $x \in \mathcal{D}_P$, on input P and x, outputs $F_P(x)$.*

We note that a particularly simple "textbook" formulation, which is useful in many interesting cases, can be obtained by setting $\mathcal{P} = \mathbb{N}$, along with the deterministic instance generation algorithm $\mathcal{IG}(1^n) = n$, and $\mathcal{D}_n = \mathcal{D}'_n = \{0,1\}^n$. However, as we will see in the following sections, for our purposes this will not be sufficient, and as a result we require the more general definition given above. In light of this, let us denote by $\mathcal{P}_n \subset \mathcal{P}$ the subset of possible parameterizations output by the instance generation algorithm $\mathcal{IG}$ on input 1^n. One thing to note from the above definition is that on input 1^n, the instance generation algorithm $\mathcal{IG}$ effectively samples from some implicitly defined distribution over $\mathcal{P}_n$ – i.e. $\mathcal{IG}(1^n)$ is a random variable taking values in $\mathcal{P}_n$. Additionally, we will assume in this work that $\mathcal{P}_n \cap \mathcal{P}_{n'} = \emptyset$ for any $n \neq n'$, and that given some $P \in \mathcal{P}_n$ it is always possible to efficiently recover n. Given this, we can define the notion of a collection of pseudorandom generators as follows.

Definition 10 (Collection of Pseudorandom Generators)**.** *An indexed collection of efficiently computable functions $\{G_P\}$ is called a collection of pseudorandom generators if for all classical probabilistic polynomial time algorithms $\mathcal{A}$, all polynomials p and all sufficiently large n it holds that*

$$\left| \Pr_{\substack{P \leftarrow \mathcal{IG}(1^n) \\ x \leftarrow U(\mathcal{D}_P)}} [\mathcal{A}(P, G_P(x)) = 1] - \Pr_{\substack{P \leftarrow \mathcal{IG}(1^n) \\ y \leftarrow U(\mathcal{D}'_P)}} [\mathcal{A}(P, y) = 1] \right| < \frac{1}{p(n)} \tag{12}$$

where $U(\mathcal{X})$ denotes the uniform distribution over the set $\mathcal{X}$, and $\mathcal{IG}$ is the instance generation algorithm for $\{G_P\}$.

We would now like to define pseudorandom functions. To do this we will need the slightly modified notion of an efficiently computable indexed collection of keyed functions:

Definition 11 (Indexed Collection of Efficiently Computable Keyed Functions)**.** *Given some infinite set of parameterizations $\mathcal{P}$, we call a collection of functions $\{F_P \mid P \in \mathcal{P}\}$ an indexed collection of efficiently computable keyed functions if for all $P \in \mathcal{P}$ we have that $F_P : \mathcal{K}_P \times \tilde{\mathcal{D}}_P \to \mathcal{D}'_P$, and there exists:*

1. *An efficient (possibly probabilistic) instance generation algorithm $\mathcal{IG}$, which on input 1^n outputs some $P \in \mathcal{P}$, as well as a description of the the key space $\mathcal{K}_P$, domain $\tilde{\mathcal{D}}_P$, and codomain $\mathcal{D}'_P$.*
2. *An efficient probabilistic key selection algorithm which on input P can sample efficiently from the uniform distribution over $\mathcal{K}_P$.*
3. *An efficient evaluation algorithm which for all $P \in \mathcal{P}$, all $k \in \mathcal{K}_P$ and all $x \in \tilde{\mathcal{D}}_P$, on input P, k, x outputs $F_P(k, x)$.*

Given this, following Refs. [32, 33], we can define various types of pseudorandom function collections via the following:

Definition 12 (Classical-Secure, Weak-Secure, Standard-Secure and Quantum-Secure Pseudorandom Function Collection)**.** *An indexed collection of efficiently computable keyed functions $\{F_P\}$ is called a (a) classical-secure (b) weak-secure (c) standard-secure or (d) quantum-secure pseudorandom function collection if for all (a,b) classical probabilistic (c,d) quantum polynomial time adversaries $\mathcal{A}$, all polynomials p, and all sufficiently large n, it holds that*

$$\left| \Pr_{\substack{P \leftarrow \mathcal{IG}(1^n) \\ k \leftarrow U(\mathcal{K}_P)}} [\mathcal{A}^{O(F_P(k,\cdot))}(P) = 1] - \Pr_{\substack{P \leftarrow \mathcal{IG}(1^n) \\ R \leftarrow U(F : \tilde{\mathcal{D}}_P \to \mathcal{D}'_P)}} [\mathcal{A}^{O(R)}(P) = 1] \right| < \frac{1}{p(n)} \tag{13}$$

where $U(F : \tilde{\mathcal{D}}_P \to \mathcal{D}'_P)$ denotes the uniform distribution over all functions from $\tilde{\mathcal{D}}_P$ to $\mathcal{D}'_P$ and $\mathcal{A}$ is given oracle access to (a,c) $O(f) = \mathrm{MQ}(f)$, (b) $O(f) = \mathrm{PEX}(f, U)$ or (d) $O(f) = \mathrm{QMQ}(f)$.

In order to clarify the above definition, we summarize informally as follows:

All efficient classical algorithms with classical random example oracle access satisfy Eq. (13) $\implies$ weak-secure.
All efficient classical algorithms with classical membership query oracle access satisfy Eq. (13) $\implies$ classical-secure.
All efficient quantum algorithms with with classical membership query oracle access satisfy Eq. (13) $\implies$ standard-secure.
All efficient quantum algorithms with quantum membership query oracle access satisfy Eq. (13) $\implies$ quantum-secure.

While at first glance the above naming conventions may seem extremely confusing, we note that if one assumes the existence of quantum computers, then the "standard" setting in which one would like to prove pseudorandomness of a function collection – i.e. the setting which corresponds to most realistic physical scenarios – is the setting in which any possible adversary (including quantum adversaries) has classical membership query access to the unknown functions [32].

3.2. Classical Hardness from Classical-Secure Pseudorandom Functions

Given the preliminaries from the previous section, we present and discuss here a construction due to Kearns et al. [27], which allows one to use (almost) any classical-secure pseudorandom function collection to construct a distribution concept class – in fact, infinitely many such classes – which are not efficiently classically learnable, with respect to the SAMPLE oracle and the KL-divergence. Before presenting this construction, however, a few remarks are necessary. Firstly, we note that Theorem 1 as presented below, is in fact a slight generalization of the original construction from Ref. [27], which makes it explicit that one can use classical-secure pseudorandom function collections parameterized by arbitrary parameterization sets (as opposed to simply $\mathcal{P} = \mathbb{N}$), provided the domain and co-domain satisfy mild requirements. This will be necessary for us, as in the following section we wish to instantiate this distribution concept class construction using a concrete pseudorandom function candidate, based on the DDH assumption. Additionally, in this work we wish to construct a distribution concept class which is not only provably hard to learn classically, but which is also provably efficiently quantum learnable. To do this we will require another modification of the Kearns construction, which is presented as Corollary 1.1, and whose significance will be discussed at length in the following section. Finally, we note that Kearns et al. have provided in Ref. [27] only a sketch of a proof that their construction yields distribution concept classes which are classically hard to learn. As we require both a slight generalization and modification (Corollary 1.1) of this construction, we provide here a full proof for Theorem 1, based on the original sketch from Ref. [27].

At this stage we are almost ready to present the construction, in a language sufficiently general for our requirements. As a final preliminary consideration, we note that for all non-negative integers $x \in \mathbb{N}^0$ we will denote by $\mathrm{BIN}(x)$ the shortest possible binary representation of x, and by $\mathrm{BIN}_n(x)$ the n-bit binary representation obtained by padding $\mathrm{BIN}(x)$ with zeros. We also denote by $x||y$ the concatenation of bit strings x and y. Additionally, for any set $X \subset \mathbb{N}^0$ we write $X \subseteq \{0,1\}^n$ when $\mathrm{BIN}_n(x)$ exists for all $x \in X$. Given these definitions we state the following theorem, a reformulation and slight generalization of the original result from Kearns et al.

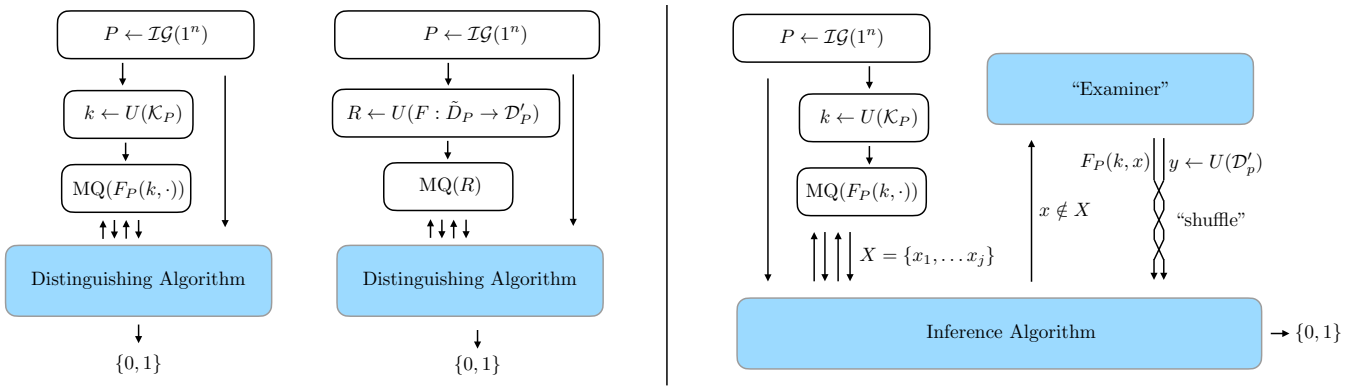

FIG. 2. Illustration of equivalent notions of a classical-secure PRF collection. The left panel illustrates the setting as per Definition 12, in which we consider distinguishing algorithms which, with membership query access, try to distinguish between a randomly drawn instance of the keyed function collection $F_P(k,\cdot)$ and a function R drawn uniformly at random. The right panel illustrates Definition 13, in which we consider an inference algorithm, which after a learning phase, should try to pass an "exam" of its own choosing. As per Lemma 1, for a given indexed collection of keyed functions, there exists a suitable distinguishing algorithm, if and only if there exists a suitable inference algorithm.

Theorem 1 (Classical Hardness from Classical-Secure Pseudorandom Functions [27]). *Let $\{F_P\}$ be a classical-secure pseudorandom function collection with the property that for all n, for all $P \in \mathcal{P}_n$, it is the case that $F_P : \mathcal{K}_P \times \{0,1\}^n \rightarrow \mathcal{D}'_P$, with $\mathcal{D}'_P \subseteq \{0,1\}^n$. For all P, and all $k \in \mathcal{K}_P$, we then define*

$$\mathsf{KGEN}_{(P,k)} : \{0,1\}^n \rightarrow \{0,1\}^{2n} \tag{14}$$

via

$$\mathsf{KGEN}_{(P,k)}(x) = x||\mathrm{BIN}_n(F_P(k,x)). \tag{15}$$

Additionally, we denote by $\tilde{D}_{(P,k)}$ the discrete distribution over $\{0,1\}^{2n}$ for which $\mathsf{KGEN}_{(P,k)}$ is a classical generator. For all sufficiently large n the distribution concept class $\tilde{\mathcal{C}}_n := \{\tilde{D}_{(P,k)} | P \in \mathcal{P}_n, k \in \mathcal{K}_p\}$ is not efficiently classically PAC learnable with respect to the SAMPLE *oracle and the KL-divergence.*

In order to simplify the presentation of the proof of Theorem 1, it will be convenient to begin with a few preliminary lemmas. The first result that we need is an alternative characterization of classical-secure pseudorandom function collections, which we develop below, and illustrate in Fig. 2.

Definition 13 (Polynomial Inference [29]). *Let $\{F_P\}$ be an indexed collection of keyed functions, and let $\mathcal{A}$ be some probabilistic polynomial time classical algorithm capable of oracle calls. On input $P \in \mathcal{P}_n$, algorithm $\mathcal{A}$ is given oracle access to $\mathrm{MQ}(F_P(k,\cdot))$, and carries out a computation in which it queries the oracle on $x_1, \ldots, x_j \in \mathcal{D}_P$. Algorithm $\mathcal{A}$ then outputs some $x \in \mathcal{D}_P$, which must satisfy $x \notin \{x_1, \ldots, x_j\}$. We call x the "exam string". At this point, $\mathcal{A}$ is then disconnected from $\mathrm{MQ}(F_P(k,\cdot))$ and presented the two values $F_P(k,x)$ and $y \leftarrow U(\mathcal{D}'_P)$ in random order. We say that $\mathcal{A}$ "passes the exam" if it correctly guesses which of the two values is $F_P(k,x)$. Let Q be some polynomial. We then say that $\mathcal{A}$ Q-infers the collection $\{F_P\}$ if for infinitely many n, given input $P \in \mathcal{P}_n$, it passes the exam with probability at least $1/2 + 1/Q(n)$, where the probability is taken uniformly over all possible choices of $P \in \mathcal{P}_n$, $k \in \mathcal{K}_P$, all possible choices of $y \in \mathcal{D}'_P$ and all possible orders of the exam strings $F_P(k,x)$ and y. We say that an indexed collection of keyed functions $\{F_P\}$ can be polynomially inferred if there exists a polynomial Q and a probabilistic polynomial time algorithm $\mathcal{A}$ which Q-infers $\{F_P\}$.*

Lemma 1 ([29]). *Let $\{F_P\}$ be an indexed collection of efficiently computable keyed functions. Then, $\{F_p\}$ cannot be polynomially inferred if and only if it is a classical-secure pseudorandom function collection.*

Additionally, we will need the following observation.

Lemma 2 ([27]). *Let GEN_D be a $(d_{\mathrm{KL}}, \epsilon)$ generator for $\tilde{D}_{(P,k)}$, for some $\epsilon < n - 2$. Then, for at least $2^n/n$ of the 2^n possible strings of the form $y = x||\mathrm{BIN}_n(F_p(k,x)) \in \{0,1\}^{2n}$ with $x \in \{0,1\}^n$ it is the case that*

$$D(x||\mathrm{BIN}_n(F_P(k,x))) \geq \frac{1}{2^{2+\epsilon+n}}. \tag{16}$$

Proof. Assume that the claim is false, i.e. that for at least $[1-(1/n)]\times 2^n$ of the strings $y = x||\mathrm{BIN}_n(F_p(k,x)) \in \{0,1\}^{2n}$ one has that

$$D(x||\mathrm{BIN}_n(F_P(k,x))) < 1/(2^{2+\epsilon+n}). \tag{17}$$

Under this assumption, and using the fact that $\log(z) \geq 0$ for all $z \geq 1$, we see that

$$\begin{aligned} d_{\mathrm{KL}}(\tilde{D}_{(P,k)}, D) &= \sum_{y\in\{0,1\}^{2n}} \tilde{D}_{(P,k)}(y) \log\left(\frac{\tilde{D}_{(P,k)}(y)}{D(y)}\right) && (18)\\ &= \sum_{x\in\{0,1\}^{n}} \frac{1}{2^n} \log\left(\frac{1}{D(x||\mathrm{BIN}_n(F_P(k,x)))}\right) - n && (19)\\ &\geq \left(1-\frac{1}{n}\right) 2^n \frac{1}{2^n} \log(2^{2+\epsilon+n}) - n && (20)\\ &= \epsilon + \left[1 - \frac{2+\epsilon}{n}\right] && (21)\\ &\geq \epsilon, && (22) \end{aligned}$$

where line (19) follows from line (18) via the observation that for all $y \in \{0,1\}^{2n}$

$$\tilde{D}_{(P,k)}(y) = \begin{cases} \frac{1}{2^n} \text{ if } y = x||\mathrm{BIN}_n(F_p(k,x)) \text{ for some } x \in \{0,1\}^n \\ 0 \text{ otherwise} \end{cases} \tag{23}$$

and line (22) follows from line (21) from the assumption that $\epsilon < n-2$. As Eq. (22) contradicts the assumption that $d_{\mathrm{KL}(D_{(P,k)},D)} \leq \epsilon$, we therefore obtain a proof of the lemma by contradiction. □

Finally, given these preliminary results and observations, we can present a full proof of Theorem 1.

Proof (Theorem 1). At a high level, the idea of the proof is to assume that $\tilde{C}_n$ is efficiently classically PAC learnable for infinitely many n, and use the associated learning algorithms to construct a poly-time algorithm which Q-infers $\{F_P\}$, for some polynomial Q. By Lemma 1 this implies that $\{F_P\}$ is not classical-secure pseudorandom, which then gives a proof by contradiction. To do this, let us denote the assumed learning algorithm for $\tilde{C}_n$ as $\tilde{\mathcal{A}}_n$. Our goal is now to construct an inference algorithm for $\{F_P\}$, which we denote as $\mathcal{A}$. Now, as per Definition 13, on input $P \in \mathcal{P}_n$, when given access to $\mathrm{MQ}(F_P(k,\cdot))$, algorithm $\mathcal{A}$ proceeds in two steps as follows:

1. Obtain an approximate generator for $\tilde{D}_{(P,k)}$ by simulating the learning algorithm $\tilde{\mathcal{A}}_n$. Specifically, run the learning algorithm $\tilde{\mathcal{A}}_n$, with $\epsilon = \log(n)$ and $\delta = 1/2$, by using access to $\mathrm{MQ}(F_P(k,\cdot))$ to simulate $\mathrm{SAMPLE}(\tilde{D}_{(P,k)})$. Each time $\tilde{\mathcal{A}}_n$ queries $\mathrm{SAMPLE}(\tilde{D}_{(P,k)})$, algorithm $\mathcal{A}$ simply draws some $x \in \{0,1\}^n$ uniformly at random, queries $\mathrm{MQ}(F_P(k,\cdot))$ on input x, and then provides $\tilde{\mathcal{A}}_n$ with the sample

$$x||\mathrm{BIN}_n[\mathsf{Query}[\mathrm{MQ}(F_P(k,\cdot)](x)] = x||\mathrm{BIN}_n(F_P(k,x)) = \mathsf{KGEN}_{(P,k)}(x). \tag{24}$$

   Let us denote by GEN_D the generator output by $\tilde{\mathcal{A}}_n$, and by $X = \{x_1,\ldots,x_j\}$ the set of strings used by $\mathcal{A}$ to simulate $\tilde{\mathcal{A}}_n$. We know that with probability $1-\delta = 1/2$, the output generator GEN_D is a $(d_{\mathrm{KL}},\epsilon)$ generator for $\tilde{D}_{(P,k)}$. Additionally, it follows from the efficiency of $\mathcal{A}_n$ that $|X| = \mathrm{poly}(n,1/\delta,1/\epsilon) = \mathrm{poly}(n)$.

2. Use GEN_D to generate a sample $x||y \in \{0,1\}^{2n}$ from D.

   - If $x \notin X$, then submit x as the exam string, and receive the strings y_1, y_2. If $y \in \{y_1,y_2\}$, then output y. Let us call this case-(a). Else, if $y \notin \{y_1,y_2\}$ then output either y_1 or y_2 uniformly at random. We call this case-(b).
   - Else, if $x \in X$, then select any $\tilde{x} \notin X$ as the exam string, and after receiving y_1, y_2 simply output either y_1 or y_2 at random. Call this case-(c).

We now want to determine a lower bound on the probability that $\mathcal{A}$ passes the exam. To do this, let us denote by $\Pr[(z)]$ the probability that case-(z) occurs, and by $\Pr_{\mathrm{pass}}[(z)]$ the conditional probability that $\mathcal{A}$ passes the exam, given that case-(z) has occurred. Clearly, $\Pr_{\mathrm{pass}}[(b)] = \Pr_{\mathrm{pass}}[(c)] = 1/2$, so let us look at case-(a) more carefully. In particular, there are two possibilities:

1. The first possibility is that $y = F_P(k, x)$. Let's call this case-(a1). In this case $\mathcal{A}$ definitely passes the exam – i.e. we have that $\Pr_{\text{pass}}[\text{(a1)}] = 1$.

2. The second possibility is that $y \neq F_P(k, x)$, and that whichever string from $\{y_1, y_2\}$ was randomly drawn, just happens to equal y. Lets call this case-(a2). In this case $\mathcal{A}$ definitely fails the exam – i.e. $\Pr_{\text{pass}}[\text{(a2)}] = 0$.

In light of this, the probability that $\mathcal{A}$ passes the exam is then given by

$$\begin{aligned}
\Pr_{\text{pass}} &= \Pr_{\text{pass}}[\text{(a1)}]\Pr[\text{(a1)}] + \Pr_{\text{pass}}[\text{(a2)}]\Pr[\text{(a2)}] + \Pr_{\text{pass}}[\text{(b)}]\Pr[\text{(b)}] + \Pr_{\text{pass}}[\text{(c)}]\Pr[\text{(c)}] \\
&= \Pr[\text{(a1)}] + \frac{1}{2}\left(\Pr[\text{(b)}] + \Pr[\text{(c)}]\right) \\
&= \Pr[\text{(a1)}] + \frac{1}{2}\left(1 - \Pr[\text{(a1)}] - \Pr[\text{(a2)}]\right) \\
&= \frac{1}{2} + \frac{1}{2}\Pr[\text{(a1)}] - \frac{1}{2}\Pr[\text{(a2)}].
\end{aligned} \tag{25}$$

So, to proceed we now analyze $\Pr[\text{(a1)}]$ and $\Pr[\text{(a2)}]$. Notice that case-(a1) occurs when $x||y = x||F_P(k, x)$ for some $x \notin X$. Using Lemma 2, with $\epsilon = \log(n)$, and n large enough that $\epsilon < n - 2$ (which would be $n > 4$) we know that, when GEN_D is a $(d_{\text{KL}}, \epsilon)$ generator for $\tilde{D}_{(P,k)}$, there exist at least $2^n/n$ strings of the form $x||F_P(k, x)$ for which

$$D(x||F_P(k, x)) \geq \frac{1}{4n2^n}. \tag{26}$$

Using the above, along with the fact that $|X| = p(n)$ for some polynomial p, we then have that

$$\begin{aligned}
\Pr[\text{(a1)}] &\geq \Pr[\text{(a1)} \,|\, d_{\text{KL}}(D, \tilde{D}_{(P,K)}) \leq \epsilon] \times \Pr[d_{\text{KL}}(D, \tilde{D}_{(P,K)}) \leq \epsilon] \\
&\geq \left[\left(\frac{1}{n}2^n - p(n)\right)\frac{1}{4n2^n}\right] \times \frac{1}{2} \\
&\geq \frac{1}{8n^2}\left(1 - np(n)/2^n\right).
\end{aligned} \tag{27}$$

As a result, there exists some n_1 such that for all $n \geq n_1$ we have that $\Pr[\text{(a1)}] \geq 1/(10n^2)$. So, at this point we know that for all n large enough

$$\Pr_{\text{pass}} \geq \frac{1}{2} + \frac{1}{10n^2} - \frac{1}{2}\Pr[\text{(a2)}]. \tag{28}$$

Now, note that case-(a2) occurs when $x \notin X$ and when whichever of y_1 or y_2 is randomly drawn is equal to y. As a result, we have that $\Pr[\text{(a2)}] \leq 1/2^n$. Using this, we see that for all $n \geq n_1$,

$$\begin{aligned}
\Pr_{\text{pass}} &\geq \frac{1}{2} + \frac{1}{10n^2} - \frac{1}{2 \times 2^n} \\
&\geq \frac{1}{2} + \frac{1}{10n^2}\left(1 - \frac{5n^2}{2^n}\right).
\end{aligned} \tag{29}$$

Similarly, to the previous case, we now know that there exists some n_2, such that for all $n \geq \max\{n_1, n_2\}$,

$$\begin{aligned}
\Pr_{\text{pass}} &\geq \frac{1}{2} + \frac{1}{11n^2} \\
&:= \frac{1}{2} + \frac{1}{Q(n)}.
\end{aligned} \tag{30}$$

In light of the above, we therefore see that for all sufficiently large n, $\mathcal{A}$ Q-infers $\{F_p\}$, and therefore, via Lemma 1, $\{F_P\}$ cannot be classical-secure pseudorandom, which contradicts the assumptions of the theorem. □

As mentioned earlier, while Theorem 1 provides a method for the construction of distribution concept classes which are not efficiently classically learnable, in order to construct such a distribution concept class which is also efficiently quantum learnable, it will be helpful to formulate the following modified construction:

Corollary 1.1. *Let $\{F_P\}$ be a classical-secure pseudorandom function satisfying all the properties required for Theorem 1. In addition, for all n, we assume that for all $P \in \mathcal{P}_n$ there exists an efficient $m = \text{poly}(n)$ bit encoding of P, which we denote as $\text{BIN}_m(P)$. For all P, and all $K \in \mathcal{K}_P$ we then define*

$$\mathsf{GEN}_{(P,k)} : \{0,1\}^n \to \{0,1\}^{2n+m} \tag{31}$$

via

$$\mathsf{GEN}_{(P,k)}(x) = x||\text{BIN}_n(F_P(k,x))||\text{BIN}_m(P). \tag{32}$$

Additionally, we define $D_{(P,k)}$ as the discrete distribution over $\{0,1\}^{2n+m}$ for which $\mathsf{GEN}_{(P,k)}$ is a classical generator. For all sufficiently large n the distribution concept class $\mathcal{C}_n := \{D_{(P,k)} | P \in \mathcal{P}_n, k \in \mathcal{K}_p\}$ is not efficiently classically PAC learnable with respect to the SAMPLE *oracle and the KL-divergence.*

To make clear the difference between the constructions of Theorem 1 and Corollary 1.1 we summarize informally as follows:

$$\begin{aligned} \text{Theorem 1} &\to \mathsf{KGEN}_{(P,k)}(x) = x||\text{BIN}_n(F_P(k,x)), \\ \text{Corollary 1.1} &\to \mathsf{GEN}_{(P,k)}(x) = x||\text{BIN}_n(F_P(k,x))||\text{BIN}_m(P). \end{aligned}$$

Before describing the motivation behind such a modification, we note that we have stated this construction as a corollary due to the fact that the proof is essentially the same as the proof of Theorem 1. The only difference is that when the polynomial inference algorithm $\mathcal{A}$ is given input $P \in \mathcal{P}_n$ and access to $\text{MQ}(F_P(k,\cdot))$, in order to simulate the learning algorithm $\tilde{\mathcal{A}}_n$, it should respond to a SAMPLE query by drawing some $x \in \{0,1\}^n$ uniformly at random, and then returning

$$x||\text{BIN}[\mathsf{Query}[\text{MQ}(F_P(k,\cdot)](x)]||\text{BIN}_m(P) = x||\text{BIN}_n(F_P(k,x))||\text{BIN}_m(P) = \mathsf{GEN}_{(P,k)}(x). \tag{33}$$

To see why such a modified construction will be helpful for constructing an efficient quantum learner, note that both $\mathsf{KGEN}_{(P,k)}$ and $\mathsf{GEN}_{(P,k)}$ are fully specified by the parameterization P, and some key $k \in \mathcal{K}_P$. As such, given SAMPLE access to either generator, it would be sufficient for a generator learning algorithm to learn the tuple (P,k). If one uses the distribution class $\tilde{D}_{(P,k)}$ of Theorem 1, generated by $\mathsf{KGEN}_{(P,k)}$, then a learner really has to learn both the parameterization P, and the key k. However, if one uses the distribution class $D_{(P,k)}$ of Corollary 1.1, generated by $\mathsf{GEN}_{(P,k)}$, then with every sample from the distribution the learner is *given* a binary encoding of P, and as such only needs to learn the key k.

3.3. Quantum Learnability and Classical Hardness for a DDH Based Distribution Concept Class

Theorem 1 and Corollary 1.1 provide the first ingredient necessary to answer Question 1 in the affirmative – namely a technique for constructing, from almost any classical-secure pseudorandom function collection, a distribution concept class which can be efficiently classically generated, and which is not efficiently classically PAC learnable. Given this result, on first impressions one might think that all that is required for such distribution concept classes to be efficiently quantum PAC learnable is that the underlying classical-secure PRF collection is not standard-secure – i.e. is not pseudorandom from the perspective of quantum adversaries with classical membership query access. In particular, it seems plausible that if the underlying PRF is classical-secure but not standard-secure, then one could exploit the quantum PRF adversary $\mathcal{A}'$ for the construction of a quantum generator learner $\mathcal{A}$. Unfortunately, however, it is in fact not so straightforward, for the following reasons: Firstly, we note that

$$\begin{aligned} \mathsf{Query}[\text{SAMPLE}(D_{(P,k)})] &= \mathsf{GEN}_{(P,k)}(x) \text{ with } x \leftarrow U_n \\ &= x||\text{BIN}_n(F_P(k,x))||\text{BIN}_m(P) \text{ with } x \leftarrow U_n. \end{aligned} \tag{34}$$

Additionally, we have that

$$\mathsf{Query}[\text{PEX}(F_P(k,\cdot),U_n)] = (x, F_P(k,x)) \text{ with } x \leftarrow U_n, \tag{35}$$

and so by comparison we see that if a learning algorithm $\mathcal{A}$ is given access to the oracle $\text{SAMPLE}(D_{(P,k)})$, then it can efficiently simulate oracle access to $\text{PEX}(F_P(k,\cdot),U_n)$, however, it *cannot* simulate oracle access to $\text{MQ}(F_P(k,\cdot))$. As such even if there exists an efficient quantum adversary $\mathcal{A}'$ for the classical-secure PRF $\{F_P\}$, a learning algorithm with oracle access to $\text{SAMPLE}(D_{(P,k)})$ could not simulate $\mathcal{A}'$, which requires access to $\text{MQ}(F_P(k,\cdot))$. Additionally, note from Eq. (13) that any quantum PRF adversary $\mathcal{A}'$ requires as input the corresponding parameterization P. As such, even if this quantum adversary could succeed with only PEX access, a learning algorithm with access only to the $\text{SAMPLE}(\tilde{D}_{(P,k)})$ oracle could not simulate $\mathcal{A}'$, as it does not have access to P. However, a learning algorithm with access to $\text{SAMPLE}(D_{(P,k)})$ could simulate $\mathcal{A}'$, as an

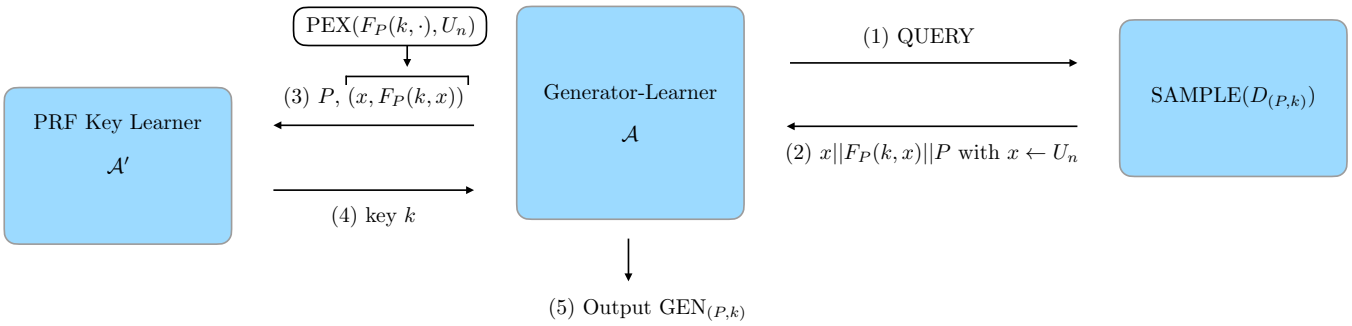

FIG. 3. Overview of the strategy for proving quantum learnability. In particular, if the classical-secure PRF $\{F_P\}$ has the property that there exists an efficient quantum *key learner*, an efficient quantum algorithm $\mathcal{A}'$ which on input P and access to $\mathrm{PEX}(F_P(k,\cdot),U_n)$ can learn the key k, then one can construct a generator-learner $\mathcal{A}$ for $\{D_{(P,k)}\}$ by using oracle access to $\mathrm{SAMPLE}(D_{(P,K)})$ to simulate the key learner $\mathcal{A}'$.

encoding of the parameterization P is given in the suffix of each sample from the distribution. This in fact provides an additional important motivation for the formulation of Corollary 1.1. Secondly, in order for a collection of keyed functions to not be standard-secure, all that is required is that a quantum adversary, with classical membership query access can, for all sufficiently large n, distinguish instances of the keyed function from randomly drawn functions with non-negligible probability, as formalized by condition (13). As such, even if the classical-secure PRF underlying the distribution concept class is not standard-secure, and even if it is not standard-secure with respect to PEX access as opposed to MQ access, this does not instantly imply that the quantum PRF adversary could be turned into a quantum distribution learner, which should for all valid (δ,ϵ) be able to learn a $(d_{\mathrm{KL}},\epsilon)$ generator, with probability $1-\delta$. However, as per the discussion in the previous section, we know that the generators $\{\mathsf{GEN}_{(P,k)}\}$ for the distribution concept class $\mathcal{C}_n = \{D_{(P,k)}\}$ are fully specified by the tuple (P,k), and that as an encoding of P is given "for free" with each sample from the distribution, being able to learn the key $k \in \mathcal{K}_P$ from SAMPLE access to $\{D_{(P,k)}\}$ is sufficient to learn the *exact* generator $\mathsf{GEN}_{(P,k)}$. Given this, we see that if the underlying classical-secure but not standard-secure PRF is such that the quantum adversary $\mathcal{A}'$

(a) requires only PEX access, as opposed to MQ access,

(b) can in addition to distinguishing an instance of the keyed function from a random function, also learn the key itself with any specified probability,

then this PRF adversary could be efficiently simulated by a learner $\mathcal{A}$ with access to $\mathrm{SAMPLE}(D_{(P,k)})$, and used to learn $\mathsf{GEN}_{(P,k)}$ with any desired probability (as illustrated in Fig. 3). Instantiating the construction of Corollary 1.1 with this PRF would then guarantee that $\{D_{(P,k)}\}$ can not be efficiently classically learned, and therefore yield a distribution concept class which provides an affirmative answer to Question 1. In light of this, we therefore in the remainder of this section construct a classical-secure but not standard-secure PRF with the properties listed above.

The PRF we construct will be classical-secure under the DDH assumption for the group family of quadratic residues. We use such a construction as this assumption is known not to hold for quantum adversaries [26]. In order to present this construction, we require the following preliminary definitions, following closely the presentation of Ref. [26]:

Definition 14 (Efficient Group Family)**.** *Given some infinite set of parameterizations $\mathcal{P}$, we say that a set of groups $\mathbb{G} = \{\mathbb{G}_P \mid P \in \mathcal{P}\}$ is an efficient group family if for all $P \in \mathcal{P}$ the group $\mathbb{G}_P$ is a finite cyclic group, and there exists:*

1. *An efficient instance generation algorithm $\mathcal{IG}$ which on input 1^n outputs some $P \in \mathcal{P}$, as well as a generator g for the group $\mathbb{G}_P$.*
2. *An efficient group product algorithm $\mathcal{G}$ which on input $P \in \mathcal{P}$ and two group elements $a, b \in \mathbb{G}_P$ outputs the group product $a * b \in \mathbb{G}_P$.*

Definition 15 (Quadratic Residues in $\mathbb{Z}_N^*$)**.** *Let $\mathbb{Z}_N^*$ denote the multiplicative group of integers modulo N. We say that an element $y \in \mathbb{Z}_N^*$ is a* quadratic residue modulo N *if there exists an $x \in \mathbb{Z}_N^*$ such that $x^2 \equiv y \mod N$. We denote the set of quadratic residues modulo N by* QR_N.

We note that QR_N forms a subgroup of $\mathbb{Z}_N^*$. Additionally, when N is prime, squaring modulo N is a two-to-one map, and therefore exactly half the elements of $\mathbb{Z}_N^*$ are quadratic residues. As a result, if N is a *safe prime* – i.e. $N = 2q+1$ with q prime – then QR_N is of prime order $q = (N-1)/2$, and therefore cyclic, with all elements (except the identity) as generators. Given this, we make the following observation:

Observation 1 (Group Family of Quadratic Residues)**.** *The set of groups $\mathbb{G} = \{\mathrm{QR}_p \,|\, p \text{ is a safe prime}\}$ is an efficient group family. The existence of an efficient group operation algorithm follows from the fact that modular multiplication can be performed efficiently [34]. Additionally, there exists (a) an efficient (probabilistic) algorithm which maps from 1^n to safe n-bit primes [34], and (b) an efficient algorithm for testing membership in QR_p [35]. Taken together, along with the observation that all elements except the identity are generators, these two algorithms allow one to construct the required efficient instance generation algorithm. We refer to $\{\mathrm{QR}_p\}$ as the group family of quadratic residues.*

With these preliminaries, we can now state the decisional Diffie-Hellman assumption:

Definition 16 (Decisional Diffie-Hellman Assumption [26, 36])**.** *We say that a group family $\mathbb{G} = \{\mathbb{G}_P\}$, with instance generator $\mathcal{IG}$, satisfies the DDH assumption, if for all probabilistic polynomial time algorithms $\mathcal{A}$, all polynomials $p(\cdot)$, and all sufficiently large n it holds that*

$$|\mathrm{Pr}_{\substack{P,g\leftarrow\mathcal{IG}(1^n)\\ a,b\leftarrow\mathbb{Z}_{|\mathbb{G}_P|}}}[\mathcal{A}(P,g,g^a,g^b,g^{ab})=1] - \mathrm{Pr}_{\substack{P,g\leftarrow\mathcal{IG}(1^n)\\ a,b,c\leftarrow\mathbb{Z}_{|\mathbb{G}_P|}}}[\mathcal{A}(P,g,g^a,g^b,g^c)=1]| < \frac{1}{p(n)}, \tag{36}$$

where the probability in the first term is taken over the random variable $\mathcal{IG}(1^n)$ (i.e. a random tuple P, g), and $a, b \in \mathbb{Z}_{|\mathbb{G}_P|}$ selected uniformly at random, and in the second term $c \in \mathbb{Z}_{|\mathbb{G}_P|}$ is also chosen uniformly at random.

From this point on we will restrict ourselves to the group family of quadratic residues parameterized by safe primes, which is believed to satisfy the DDH assumption [26]. The first thing to note is that the DDH assumption instantly implies a method for the construction of a collection of pseudorandom generators. To show this, we begin by defining the function $\mathsf{modexp}_{p,g} : \mathbb{N} \to \mathrm{QR}_p$ via

$$\mathsf{modexp}_{p,g}(b) := g^b \bmod(p). \tag{37}$$

Given this, we then define an indexed collection of functions $\{G_{(p,g,g^a)}\}$, which is parameterized by a tuple (p, g, g^a), where $p = 2q + 1$ is some safe-prime, g is generator for QR_p and $a \in \mathbb{Z}_q$. We denote the infinite set of such valid parameterizations as $\mathcal{P}_{(p,g,g^a)}$, and the subset of all such parameterizations in which p is an n-bit prime by $\mathcal{P}_{n,(p,g,g^a)}$. Note that on input 1^n the instance generator $\mathcal{IG}$ for this indexed collection first runs the instance generator for $\{\mathrm{QR}_p\}$, outputting a tuple p, g, and then selects some $a \in \mathbb{Z}_q$ uniformly at random. More specifically, for all valid parameterizations $(p, g, g^a) \in \mathcal{P}_{(p,g,g^a)}$ we have that

$$G_{(p,g,g^a)} : \mathbb{Z}_q \to \mathrm{QR}_p \times \mathrm{QR}_p \tag{38}$$

via

$$\begin{aligned} G_{(p,g,g^a)}(b) &= \mathsf{modexp}_{p,g}(b)||\mathsf{modexp}_{p,g^a}(b) \\ &= g^a \bmod(p)||g^{ab} \bmod(p) \\ &:= G^0_{(p,g,g^a)}(b)||G^1_{(p,g,g^a)}(b). \end{aligned} \tag{39}$$

Given this construction, we now we note, following Ref. [26], that $\{G_{(p,g,g^a)}\}$ is a collection of pseudorandom generators, under the assumption that $\{\mathrm{QR}_p\}$ satisfies the DDH assumption.

Observation 2 ($\{G_{(p,g,g^a)}\}$ is a Collection of Pseudorandom Generators [26])**.** *Note that when using the group family of quadratic residues, we can rewrite Eq. (36) as*

$$|\mathrm{Pr}_{\substack{p,g,g^a\leftarrow\mathcal{IG}(1^n)\\ b\leftarrow\mathbb{Z}_q}}[\mathcal{A}(p,g,g^a,G_{(p,g,g^a)}(b))=1] - \mathrm{Pr}_{\substack{p,g,g^a\leftarrow\mathcal{IG}(1^n)\\ b,c\leftarrow\mathbb{Z}_q}}[\mathcal{A}(P,g,g^a,g^b,g^c)=1]| < \frac{1}{p(n)}. \tag{40}$$

By comparison with Definition 10, we therefore see that $\{G_{(p,g,g^a)}\}$ is an indexed collection of pseudorandom generators, under the assumption that $\{\mathrm{QR}_p\}$ satisfies the DDH assumption.

We would now like to use the PRG $\{G_{(p,g,g^a)}\}$ to define a suitable classical-secure PRF collection. As $\{G_{(p,g,g^a)}\}$ is a collection of effectively length doubling pseudorandom generators, it has been observed multiple times [26, 33, 36] that one could in principle construct a classical-secure PRF collection from $\{G_{(p,g,g^a)}\}$ via the Goldreich-Goldwasser-Micali (GGM) construction [29, 31]. However, as noted and discussed in Ref. [37], one needs to take some care in order to construct such a PRF collection in a rigorous way. More specifically, the GGM construction requires that the functions $G^0_{(p,g,g^a)}$ and $G^1_{(p,g,g^a)}$ (defined via Eq. (39)) be iterated in an order defined via the input to the PRF, and for such an iteration to be well defined, and for the GGM proof to hold with only minor modifications, it is essential that there exists an efficient bijection from the codomain of $G^i_{(p,g,g^a)}$ to its domain – i.e. a function which efficiently enumerates the elements of the group. However, for the group family of quadratic residues, such an

efficient bijection exists, and as such it is indeed possible to construct a PRF collection, via the GGM construction, starting from a slightly modified DDH based PRG collection. To make this more precise, given some safe prime $p = 2q + 1$, we define the function $f_p : \mathrm{QR}_p \to \mathbb{Z}_q$ via

$$f_p(x) = \begin{cases} x & \text{if } x \leq q, \\ p - x & \text{if } x > q. \end{cases} \tag{41}$$

As noted in Ref. [37], this function is fact a bijection, whose inverse $f_p^{-1} : \mathbb{Z}_q \to \mathrm{QR}_p$ is given by

$$f_p^{-1}(y) = \begin{cases} y & \text{if } y \in \mathrm{QR}_p, \\ p - y & \text{if } y \notin \mathrm{QR}_p. \end{cases} \tag{42}$$

While it is clear that f_p can be efficiently computed, efficiency of f_p^{-1} is less obvious, and follows from the fact that group membership in QR_p can be efficiently tested [35]. With this in hand, we now define the indexed collection of functions $\{\tilde{G}_{(p,g,g^a)}\}$, where for all valid parameterizations

$$\tilde{G}_{(p,g,g^a)} : \mathbb{Z}_q :\to \mathbb{Z}_q \times \mathbb{Z}_q \tag{43}$$

is defined via

$$\begin{aligned} \tilde{G}_{(p,g,g^a)} &= f_p(G^0_{(p,g,g^a)}(b))||f_p(G^1_{(p,g,g^a)}(b)) \\ &:= \tilde{G}^0_{(p,g,g^a)}(b)||\tilde{G}^1_{(p,g,g^a)}(b). \end{aligned} \tag{44}$$

As f_p is a bijection, we again have, analogously to Observation 2, that $\{\tilde{G}_{(p,g,g^a)}\}$ is an indexed collection of PRG's, under the DDH assumption for $\{\mathrm{QR}_p\}$. Given this, we can finally construct the DDH based PRF which will fulfill all our requirements. Specifically, we consider an indexed collection of keyed functions $\{F_{(p,g,g^a)}\}$, where

$$F_{(p,g,g^a)} : \mathbb{Z}_q \times \{0,1\}^n \to \mathbb{Z}_q \tag{45}$$

is defined algorithmically, via the GGM construction [29, 31], as follows:

Algorithm 1 Algorithmic implementation of $F_{(p,g,g^a)}(b, x)$

Given parameterization p, g, g^a, as well as key $b \in \mathbb{Z}_q$ and input $x \in \{0,1\}^n$
$b_0 \leftarrow b$
for all $1 \leq j \leq n$ **do**
  if $x_j = 0$ **then**
    $b_j \leftarrow \tilde{G}^0_{(p,g,g^a)}(b_{j-1}) = f_p(\mathsf{modexp}_{p,g}(b_{j-1})) = f(g^{b_{j-1}} \bmod(p))$
  else if $x_j = 1$ **then**
    $b_j \leftarrow \tilde{G}^1_{(p,g,g^a)}(b_{j-1}) = f_p(\mathsf{modexp}_{p,g^a}(b_{j-1})) = f(g^{ab_{j-1}} \bmod(p))$
  end if
end for
Output: $b_n \in \mathbb{Z}_q$ ▷ Note that $b_n = \tilde{G}^{x_n}_{(p,g,g^a)}(\ldots(\tilde{G}^{x_1}_{(p,g,g^a)}(b)))$

The above algorithm is also illustrated in Fig. 4, which serves to illustrate that for all tuples (b, x) the desired output $F_{(p,g,g^a)}(b, x)$ can be calculated by moving through a binary tree, with the key $b \in \mathbb{Z}_q$ at the root, and where at each level either $\tilde{G}^0_{(p,g,g^a)}$ or $\tilde{G}^1_{(p,g,g^a)}$ is applied, with the path determined by the input string $x \in \{0,1\}^n$. We now make the following claims:

Claim 1. *$\{F_{(p,g,g^a)}\}$ is a classical-secure pseudorandom function collection.*

Claim 2. *For all $n \in \mathbb{N}$, there exists an exact efficient quantum key-learning algorithm which, for all valid parameterizations $(p, g, g^a) \in \mathcal{P}_{n,(p,g,g^a)}$, all $x \in \{0,1\}^n$, and all $b \in \mathbb{Z}_q$, on input (p, g, g^a), x and $F_{(p,g,g^a)}(b, x)$ returns b with probability 1.*

Before proceeding to prove these claims, we make the following observations: Firstly, note that given Claim 1, the indexed collection of keyed functions $\{F_{(p,g,g^a)}\}$ satisfies all the requirements of both Theorem 1 and Corollary 1.1, and as such, for all sufficiently large n, the distribution concept class

$$\mathcal{C}_n = \{D_{(p,g,g^a),b} \,|\, (p, g, g^a) \in \mathcal{P}_{n,(p,g,g^a)}, b \in \mathbb{Z}_q\}, \tag{46}$$

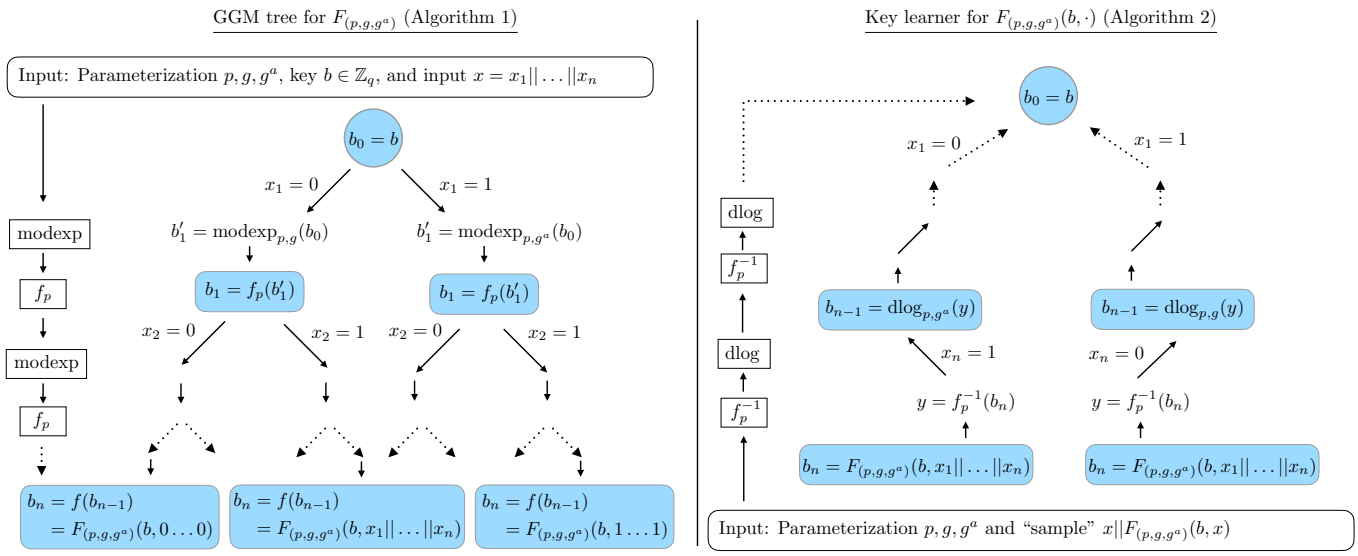

FIG. 4. (Left Panel) An illustation of the GGM tree construction for $\{F_{(p,g,g^a)}\}$. For all input tuples (b, x) the desired output $F_{(p,g,g^a)}(b, x)$ can be calculated by moving through a binary tree, with the key $b \in \mathbb{Z}_q$ at the root, and where at each level either $\tilde{G}^0_{(p,g,g^a)} = f_p \circ \mathsf{modexp}_{p,g}$ or $\tilde{G}^1_{(p,g,g^a)} = f_p \circ \mathsf{modexp}_{p,g^a}$ is applied, with the path determined by the input string $x \in \{0,1\}^n$. (Right Panel) An illustration of the "tree-reversal" key-learning algorithm, which given a leaf of the tree $F_{(p,g,g^a)}(b, x)$, along with a description of the path x from the root to this leaf, exploits the knowledge of this path, along with the ability to invert $\tilde{G}^i_{(p,g,g^a)}$, to reverse the tree and obtain the value of the root node.

defined as per Corollary 1.1, is not efficiently classically PAC learnable with respect to SAMPLE. Additionally, let us assume that a quantum generator-learner $\mathcal{A}$ is given access to $\text{SAMPLE}(D_{(p,g,g^a),b})$, for some $(p, g, g^a) \in \mathcal{P}_{n,(p,g,g^a)}$. Note that

$$\mathsf{Query}[\text{SAMPLE}(D_{(p,g,g^a),b})] = x||\text{BIN}(F_{(p,g,g^a)}(b, x))||\text{BIN}(p, g, g^a) \text{ with } x \leftarrow U_n \tag{47}$$

and therefore, as illustrated in Fig. 3, from any single such sample, the quantum generator-learner $\mathcal{A}$ can run the quantum key-learning algorithm $\mathcal{A}'$ from Claim 2, on input (p, g, g^a), x and $F_{(p,g,g^a)}(b, x)$, and obtain the key b with probability 1, which coupled with the parameterization (p, g, g^a) provides a complete description of the exact generator $\mathsf{GEN}_{((p,g,g^a),b)}$ for $D_{(p,g,g^a),b}$. As a result, it follows from Claim 2 that the distribution concept class $\{D_{(p,g,g^a),b}\}$ is efficiently quantum PAC learnable with SAMPLE access. Given this discussion, we see that the following theorem, providing an affirmative answer to Question 1, follows from Claims 1 and 2:

Theorem 2 (Quantum Generator-Learning Advantage for $\{D_{(p,g,g^a),b}\}$)**.** *For all sufficiently large n the distribution concept class*

$$\mathcal{C}_n = \{D_{(p,g,g^a),b} \,|\, (p, g, g^a) \in \mathcal{P}_{n,(p,g,g^a)}, b \in \mathbb{Z}_q\} \tag{48}$$

is not efficiently classically PAC generator-learnable with respect to the SAMPLE *oracle and the KL-divergence, but is efficiently quantum PAC learnable with respect to the* SAMPLE *oracle and the KL-divergence.*

At this point all that remains is to provide the proofs of Claims 1 and 2. As $\{\tilde{G}_{(p,g,g^a)}\}$ is a collection of length doubling PRG's the proof of Claim 1 is essentially identical to the original GGM proof [29, 31], with only minor straightforward modifications required to the definition of the hybrid distributions in order to account for the slightly more general parameterization set. As such, we omit this proof and proceed with a constructive proof of Claim 2. Intuitively, as illustrated in Fig. 4, given (p, g, g^a), x and $F_{(p,g,g^a)}(b, x)$ – i.e. a "leaf" of the GGM tree – we construct an algorithm which can reverse the path taken through the tree to reach this leaf, and return the value of the root node, which is the desired key. More formally, the proof is as follows:

Proof (Claim 2). For all tuples p, g, where $p = 2q + 1$ is an n-bit safe prime, and g is a generator for QR_p, we define the discrete log function $\mathsf{dlog}_{p,g} : \text{QR}_p \to \mathbb{Z}_q$ via

$$\mathsf{dlog}_{p,g}(g^x) = x \bmod(q), \tag{49}$$

for all $x \in \mathbb{N}$. As shown in Ref [28], for all relevant tuples (p, g), there exists an exact efficient quantum algorithm for computing $\mathsf{dlog}_{p,g}$ – i.e. an efficient quantum algorithm which on input p, g, g^x outputs $\mathsf{dlog}_{p,g}(g^x)$ with probability 1. Given the ability to efficiently calculate dlog as a subroutine, we can now construct a "tree-reversal" algorithm by using the string x to determine

whether $f_p \circ \mathsf{modexp}_{p,g}$ or $f_p \circ \mathsf{modexp}_{p,g^a}$ was applied at a given level of the tree, and then applying either $\mathsf{dlog}_{p,g} \circ f_p^{-1}$ or $\mathsf{dlog}_{p,g^a} \circ f_p^{-1}$ as appropriate. More specifically, the following algorithm, also illustrated in Fig. 4, provides a constructive proof for the claim:

Algorithm 2 Exact efficient quantum key learner for $F_{p,g,g^a}(b,x)$

Given parameterization (p, g, g^a), as well as a "sample" $x||F_{(p,g,g^a)}(b,x)$
$b_n \leftarrow F_{(p,g,g^a)}(b,x)$
for all $n \geq j \geq 1$ **do**
    $y \leftarrow f_p^{-1}(b_j)$
    if $x_j = 0$ **then**
        $b_{j-1} \leftarrow \mathrm{dlog}_{p,g}(y)$
    else if $x_j = 1$ **then**
        $b_{j-1} \leftarrow \mathrm{dlog}_{p,g^a}(y)$
    end if
end for

Output: $b_0 \in \mathbb{Z}_q$

□

4. ON CLASSICAL HARDNESS RESULTS FROM ALTERNATIVE PRIMITIVES

One of the key tools which allowed us to prove our main result (Theorem 2) was the construction and associated classical hardness result of Kearns et al. [27], presented here as Theorem 1. In this section we shift direction slightly, and explore the possibility of obtaining similar classical hardness results from alternative primitives. More specifically, as we have seen in the previous section, Kearns et al. have shown that given any classical-secure PRF collection, it is possible to construct a distribution concept class which is not efficiently classically learnable. In Section 4.1 we ask whether it is necessary to use a PRF collection to obtain such a classical hardness result, or whether the weaker notion of a *weak*-secure PRF collection [33] would be sufficient. This question is motivated partly by the existence of weak-secure PRF's with known quantum adversaries [33], upon which, if one is able to obtain a classical hardness result, then one might be able to obtain additional quantum/classical distribution learning separations, which are possibly achievable by near-term quantum learners. In Section 4.2 we then ask whether one could instantiate the construction of Kearns et al. with a *Boolean function* concept class which is provably not efficiently classically PAC learnable. In this latter case, the motivation is to understand better the relationship between PAC learning of Boolean functions and PAC learning of discrete distributions, and in particular whether one could leverage existing classical/quantum separations for learning Boolean functions into distribution learning separations. In Sections 4.1 and 4.2 we formulate conjectures concerning the possibility of using these alternative primitives for classical hardness results, and describe clearly some obstacles towards proving these conjectures.

4.1. Classical Hardness Results from Weak-Secure Pseudorandom Function Collections

Let us begin by discussing the possibility of proving a classical/quantum learning separation that is based on weak-secure PRFs as opposed to classical-secure PRFs, as they are used in Theorem 1. In order to understand the motivation for this question, it is necessary to briefly return to an analysis of the proof of Theorem 1. In particular, at an informal level (i.e. modulo some details) we recall the following:

1. Theorem 1 states that if $\{F_P\}$ is a classical-secure PRF collection, then for infinitely many n the distribution concept class $\tilde{\mathcal{C}}_n = \{D_{(P,k)}\}$ is not classically efficiently PAC generator-learnable.

2. The proof of Theorem 1 proceeds by assuming that the concept class $\tilde{C}_n$ is efficiently PAC learnable for infinitely many n, and then using the associated learning algorithms $\tilde{\mathcal{A}}_n$ to construct a polynomial time algorithm $\mathcal{A}$ which Q-infers the function collection $\{F_P\}$, for some polynomial Q, thereby contradicting the assumption that $\{F_P\}$ is a classical-secure pseudorandom function collection.

More specifically, the proof of Theorem 1 begins with the observation that

$$\begin{aligned}\mathsf{Query}[\mathrm{SAMPLE}(\tilde{D}_{(P,k)})] &= \mathsf{KGEN}_{(P,k)}(x) \text{ with } x \leftarrow U_n \\ &= x||\mathrm{BIN}_n(F_P(k,x)) \text{ with } x \leftarrow U_n \\ &= x||\mathrm{BIN}_n(\mathsf{Query}[\mathrm{MQ}(F_P(k,\cdot))](x)) \text{ with } x \leftarrow U_n \end{aligned} \tag{50}$$

and therefore when given oracle access to $\mathrm{MQ}(F_P(k,\cdot))$, the polynomial inference algorithm $\mathcal{A}$ can simulate the learner $\tilde{\mathcal{A}}_n$ by responding to sample queries of $\tilde{\mathcal{A}}_n$ with $x||\mathrm{BIN}_n(\mathsf{Query}[\mathrm{MQ}(F_P(k,\cdot))](x))$, where x has been drawn uniformly at random. Algorithm $\mathcal{A}$ then uses the obtained generator to "pass the exam" described in Definition 13. However, we note that in order to simulate the learner $\tilde{\mathcal{A}}$, it is *not* necessary for the inference algorithm to have membership query access to $F_P(k,\cdot)$. In particular, we have that

$$\mathsf{Query}[\mathrm{SAMPLE}(\tilde{D}_{(P,k)})] = x||\mathrm{BIN}_n(\mathsf{Query}[\mathrm{PEX}(F_P(k,\cdot),U)]) \tag{51}$$

and therefore the entire proof of Theorem 1 holds, even if the polynomial inference algorithm $\mathcal{A}$ only has *random example* oracle access to $F_P(k,\cdot)$. In light of this observation, one may immediately think that a *weak*-secure PRF collection would be sufficient for instantiating the Kearns construction described in Theorem 1. In other words, given that the proof of Theorem 1 holds when the polynomial inference algorithm only has random example oracle access to $F_P(k,\cdot)$, it seems plausible that if one can efficiently learn the distribution concept class $\{\tilde{\mathcal{D}}_{P,k}\}$, then one can use this learner to construct an adversary (i.e. distinguishing algorithm) for the function collection $\{F_P\}$ which only requires random example oracle access. To formalize this idea, we make the following conjecture, a slight variant of Theorem 1:

Conjecture 1. *Let $\{F_P\}$ be a weak-secure pseudorandom function collection with the additional properties stated in Theorem 1. Then, for all sufficiently large n the distribution concept class $\tilde{\mathcal{C}}_n := \{\tilde{D}_{(P,k)} | P \in \mathcal{P}_n, k \in \mathcal{K}_p\}$ is not efficiently classically PAC generator-learnable with respect to the* SAMPLE *oracle and the KL-divergence.*

Given the proof of Theorem 1, and the observation that the polynomial inference algorithm involved in the proof requires only random example oracle access, one might think that a proof of Conjecture 1 would follow immediately. Unfortunately though, this is not the case. However, before discussing the obstacles one faces in adapting the proof of Theorem 1 to prove Conjecture 1, it is worth examining briefly why Conjecture 1 is interesting, and what consequences its truth might have for obtaining classical/quantum distribution learning separations. Firstly, as discussed in Ref. [33] there is currently strong evidence that weak-secure PRF's are indeed a less complex object than classical-secure PRF's. More specifically, it is believed that there exist weak-secure PRF's which are not classical-secure PRF's [33], and as such Conjecture 1 would provide evidence that the existence of classical-secure PRF's is not necessary for the construction of distribution concept classes which are provably not classically efficiently PAC generator-learnable. Additionally one candidate for such a weak-secure PRF collection is based on the "learning parity with noise" problem[2], which is strongly believed to be hard for classical algorithms with classical random examples [38], but which is known to be efficiently solvable by quantum algorithms with *quantum* random examples [39, 40]. Importantly, unlike quantum algorithms for the discrete logarithm [28], which seem to require a universal fault-tolerant quantum computer, quantum algorithms for "learning parity with noise" (LPWN) are robust against certain types of noise models [39], and have in fact already been demonstrated on existing NISQ devices [41]. As such, while demonstrating a quantum advantage for the generator-learnability of the DDH based concept class described in Section 3.3 would require a universal fault-tolerant quantum computer, it is plausible that if Conjecture 1 is true, then one could construct a LPWN based concept class which is not classically efficiently PAC generator-learnable, but which is quantum efficiently PAC generator learnable using existing or near-term quantum devices, albeit with quantum random samples.

With these observations in mind, let us return to a discussion of the difficulties in adapting the proof of Theorem 1 into a proof for Conjecture 1. Analogously to the proof of Theorem 1, we would like to show that if the distribution concept class $\{\tilde{D}_{(P,k)}\}$ is efficiently PAC-generator learnable (with respect to SAMPLE and the KL-divergence) then the function collection $\{F_P\}$ is not weak-secure. However, it is critical to note that in proving Theorem 1 we relied heavily on the alternative characterization of classical-secure PRF's provided by Lemma 1. More specifically, we used the fact that if there exists a polynomial inference algorithm (using membership queries) for an indexed collection of keyed functions, then this collection of functions is not standard-secure. Now, from the previous discussion, we know that if the distribution concept class $\{\tilde{D}_{(P,k)}\}$ is efficiently PAC-generator learnable, then there exists an efficient polynomial inference algorithm for $\{F_P\}$ which only requires random examples, as opposed to membership queries. However, it is *not* clear that this implies that the function collection $\{F_P\}$ is not weak-secure! In other words, in order to adapt the proof of Theorem 1 to a proof of Conjecture 1 we need an alternative characterization of weak-secure PRF's analagous to Lemma 1 – i.e. a statement that if there exists an efficient polynomial inference algorithm for

[2] We note that this weak-secure PRF collection is in fact a *randomized* function collection – i.e. the evaluation algorithm is allowed to be probabilistic – however we leave out a discussion of this subtlety, and refer to Ref. [33] for more details.

$\{F_P\}$ which only requires random examples, then $\{F_P\}$ is not a weak-secure PRF collection. To understand why obtaining such a characterization is tricky, it is necessary to sketch the original proof of Lemma 1 from Ref. [29]. In order to prove the direction that we are concerned with, one starts by assuming that there exists a polynomial inference algorithm $\mathcal{A}$ for $\{F_P\}$, and then using this algorithm to construct a new distinguishing algorithm $\mathcal{A}'$ which, when given membership query access to some unknown function F can with non-negligible probability determine whether this function was drawn uniformly at random from the set of all functions $F : \tilde{D}_P \to \mathcal{D}'_P$, or uniformly at random from the set of functions $\{F_P(k, \cdot) \,|\, k \in \mathcal{K}_P\}$. More specifically, algorithm $\mathcal{A}'$ works as follows:

1. When given some parameterization P, along with oracle access to $\mathrm{MQ}(F)$, the distinguishing algorithm $\mathcal{A}'$ begins by simulating the inference algorithm $\mathcal{A}$, which returns an "exam string" x.
2. Using $\mathrm{MQ}(F)$, algorithm $\mathcal{A}'$ then "prepares the exam" – i.e. presents algorithm $\mathcal{A}$ with $y_1 = F(x)$ and $y_2 \leftarrow U_{\mathcal{D}'_P}$ in a random order.
3. The inference algorithm $\mathcal{A}$ then "takes the exam", and picks either y_1 or y_2.
4. If $\mathcal{A}$ picks y_1, then $\mathcal{A}'$ outputs 1, otherwise $\mathcal{A}'$ outputs 0.

The fact that $\mathcal{A}$ succeeds with non-neglible probability then follows straightforwardly from the fact that $\mathcal{A}'$ Q-infers $\{F_P\}$ for some polynomial Q [29]. In light of the above sketch, we can now analyze the difficulties one faces in adapting the above proof when both the distinguishing algorithm and the inference algorithm are only allowed random example oracle access. Recall, we want to show that if $\mathcal{A}$ Q-infers $\{F_P\}$ using only random examples – i.e. with PEX access – then we can build a suitable distinguishing algorithm $\mathcal{A}'$, which also only requires random examples (which would imply $\{F_P\}$ is not weak-secure). So, as per the proof sketched above for the case of membership queries, when given access to $\mathrm{PEX}(F, U)$, the distinguishing algorithm $\mathcal{A}'$ could start by simulating the inference algorithm $\mathcal{A}$, which returns an exam string x. It is at this point that we encounter a problem! Specifically, given the exam string x, $\mathcal{A}'$ should prepare the exam by returning $y_1 = F(x)$ along with some y_2 drawn uniformly at random from $\mathcal{D}'_P$. If $\mathcal{A}'$ has access to $\mathrm{MQ}(x)$ then it is straightforward to prepare the exam, as $\mathsf{Query}[\mathrm{MQ}(F)](x) = F(x)$. However, if $\mathcal{A}'$ only has access to $\mathrm{PEX}(F, U)$, then it *cannot* prepare the exam! Note that if we modified the definition of polynomial inference (given as Definition 13) so that the inference algorithm does not get to choose its exam string, but is just given an exam string sampled uniformly at random (from the set of strings which have not yet been used), then algorithm $\mathcal{A}'$ *could* prepare the exam for algorithm $\mathcal{A}$, and the rest of the proof would hold, yielding an alternative characterization of weak-secure PRF collections in terms of a slightly modified notion of polynomial inference. However, we note that with such a modified definition of polynomial inference, the proof of Theorem 1 will no longer work! In particular, recall that the proof of Theorem 1 relies heavily on the fact that the constructed inference algorithm can use the generator it obtained from the distribution learner to *choose* its own exam string. In other words, if a polynomial inference algorithm for $\{F_P\}$ is required to pass a randomly drawn exam with non-negligible probability, then it is completely unclear how a distribution learner for $\{D_{(P,k)}\}$ can be used to construct a successful polynomial inference algorithm. Given these observations we see that, while Conjecture 1 seems plausible, and has a variety of interesting consequences if true, one cannot simply adapt the proof of Theorem 1 to this modified setting.

4.2. Classical Hardness Results from Hard to Learn Function Concept Classes

While in this work we have so far focused primarily on the PAC learnability of *distribution* concept classes, as an abstraction of generative modelling, there exists already a large body of work concerning the quantum versus classical PAC learnability of *Boolean function* concept classes [4]. In this section, we aim to explore to some extent the relationship between these two notions, and in particular whether existing results in the latter context can be leveraged to obtain results in the former. As a starting point, we note that in principle one could instantiate the distribution class construction from Kearns et al. [27] with a Boolean function concept class, as formalized by the following definition:

Definition 17. *Given some Boolean function $f \in \mathcal{F}_n$, we define the distribution $D_f \in \mathcal{D}_{n+1}$ as the distribution defined via the classical generator $\mathsf{GEN}_{D_f} : \{0,1\}^n \to \{0,1\}^{n+1}$ built from f via*

$$\mathsf{GEN}_{D_f}(x) = x||f(x). \tag{52}$$

Additionally, given some concept class $\mathcal{C} \subseteq \mathcal{F}_n$ we define the distribution class $\mathcal{D}_\mathcal{C} \subseteq \mathcal{D}_{n+1}$ via

$$\mathcal{D}_\mathcal{C} = \{D_c \,|c \in \mathcal{C}\}. \tag{53}$$

Given the above construction, we proceed in this section to prove Theorem 3, and to discuss in detail its inverse statement, which we formalize as Conjecture 2.

Theorem 3 (Function Learnability Implies Distribution Learnability)**.** *If a concept class $\mathcal{C}$ is efficiently classically (quantum) PAC learnable with respect to the uniform distribution and the* PEX *oracle, then the distribution concept class $\mathcal{D}_{\mathcal{C}}$ is efficiently classically (quantum) PAC generator-learnable with respect to the* SAMPLE *oracle and the TV distance.*

Conjecture 2 (Function Hardness Implies Distribution Hardness)**.** *If a concept class $\mathcal{C}$ is not efficiently classically (quantum) PAC learnable with respect to the uniform distribution and the* PEX *oracle, then the distribution concept class $\mathcal{D}_{\mathcal{C}}$ is not efficiently classically (quantum) PAC generator-learnable with respect to the* SAMPLE *oracle and the TV distance.*

Apart from shedding some light on the relationship between function learnability and distribution learnability, what we might hope is that taken together Theorem 3 and Conjecture 2 (if true) would allow us to instantly leverage some existing separation between the classical versus quantum learnability of a particular Boolean function concept class $\mathcal{C}$, to obtain a separation between the classical versus quantum learnability of the associated distribution concept class $\mathcal{D}_{\mathcal{C}}$. Unfortunately however this is not the case. In particular, we stress that both Theorem 3 and Conjecture 2 describe a relationship between the generator-learnability of $\mathcal{D}_{\mathcal{C}}$, and the *distribution specific* PAC learnability of the concept class $\mathcal{C}$, *with respect to the uniform distribution*, as well as with respect to the *classical* random example PEX oracle. More specifically, what this means is that, if Conjecture 2 is true, and if there exists a concept class $\mathcal{C}$ which has the following properties:

(a) $\mathcal{C}$ *is not* efficiently *classically* PAC learnable, *with respect to the uniform distribution* and the PEX oracle,

(b) $\mathcal{C}$ *is* efficiently *quantum* learnable, *with respect to the uniform distribution* and the PEX oracle,

then the distribution class $\mathcal{D}_{\mathcal{C}}$ would not be efficiently classically PAC generator-learnable (via Conjecture 2), but it would be efficiently quantum PAC-generator learnable (via Theorem 3). However, at present it is not known whether a concept class $\mathcal{C}$ with both of the above properties exists. More specifically, as discussed in Ref. [4], Kearns and Valiant [42] have constructed a concept class which, under the assumption that there exists no efficient algorithm for the factorization of Blum integers, is not efficiently PAC learnable with respect to the PEX oracle, but which Servedio and Gortler [43] have shown is efficiently quantum PAC learnable with respect to the PEX oracle. However, recall that in order to prove that a concept class is not efficiently PAC learnable, all one has to do is prove that there exists a single distribution D with respect to which the concept class is not efficiently PAC learnable. As such, it can be that a concept class is not efficiently PAC learnable, while still being efficiently PAC learnable *with respect to the uniform distribution* – which is the case for the factoring based concept class of Kearns and Valiant.

If one restricts themselves to PAC learnability with respect to the uniform distribution, Bshouty and Jackson [9] have shown that the concept class of s-term DNF, whose best known classical learner with PEX access requires quasi-polynomial time [44], is efficiently quantum PAC learnable, if one allows the learner access to the *quantum* random example oracle QPEX. As such we see that the factoring based concept class of Kearns and Valiant fails to satisfy our requirements due to the fact that it is efficiently classically PAC learnable with respect to the uniform distribution, while the concept class of s-term DNF fails to satisfy our requirements due to the fact that the efficient quantum learner requires quantum random examples. Despite this we note that Kharitonov [45, 46] has given a variety of concept classes, which under various cryptographic assumptions, satisfy property (a) above – i.e. are not efficiently classically PAC learnable with respect to the uniform distribution and the random example oracle[3]. In light of these results, we see that the the truth of Conjecture 2 would at the least imply the existence of a distribution concept class which is not efficiently classically PAC generator-learnable. Given these observations, we proceed to prove Theorem 3, and to discuss in more detail Conjecture 2.

In order to prove Theorem 3 we begin with a few preliminary results and observations. The first such observation follows directly from Definition 4:

Observation 3. *Let $\mathsf{GEN}_D : \{0,1\}^m \to \{0,1\}^n$ be a classical generator for some probability distribution $D \in \mathcal{D}_n$. Then, for all $y \in \{0,1\}^n$ we have that*

$$D(y) = \frac{1}{2^m} \sum_{x \in \{0,1\}^m} \delta(\mathsf{GEN}_D(x), y), \tag{54}$$

where $\delta(y', y) = 1$ if $y = y'$ and $\delta(y', y) = 0$ otherwise.

The above observation then allows us to prove the following lemma:

Lemma 3. *For any two Boolean functions $h, c \in \mathcal{F}_n$, we have that*

$$\Pr_{x \leftarrow U_n}[h(x) \neq c(x)] = d_{\mathrm{TV}}(D_h, D_c) \tag{55}$$

---

[3] We note that Kharitonov's results [45, 46] are in fact significantly stronger. In particular, he provides concept classes which are not even *weakly* learnable (i.e. with non-negligible advantage) with respect to the uniform distribution, even if the learner is allowed membership queries.

Proof. Firstly, note that it follows from Eq. (54) that for any Boolean function $f \in \mathcal{F}_n$, we have for all $y \in \{0,1\}^{n+1}$ that

$$\begin{aligned} D_f(y) &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \delta(\mathsf{GEN}_{D_f}(x), y) \\ &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} \delta\Big(x||f(x), y_{[1,n]}||y_{n+1}\Big) \\ &= \frac{1}{2^n} \delta\Big(f(y_{[1,n]}), y_{n+1}\Big), \end{aligned} \tag{56}$$

where we have denoted the n bit prefix of y with $y_{[1,n]}$ and the $(n+1)$'th bit of y with y_{n+1}. Given this, we then have that

$$\begin{aligned} d_{\mathrm{TV}}(D_h, D_c) &= \frac{1}{2} \sum_{y \in \{0,1\}^{n+1}} |D_h(y) - D_c(y)| \\ &= \frac{1}{2} \left(\frac{1}{2^n} \sum_{y \in \{0,1\}^{n+1}} \left| \delta\Big(h(y_{[1,n]}), y_{n+1}\Big) - \delta\Big(c(y_{[1,n]}), y_{n+1}\Big) \right| \right) \\ &= \frac{1}{2} \left(\frac{1}{2^n} \sum_{y \in \{0,1\}^{n}} 2[1 - \delta(h(y), c(y))] \right) \\ &= \Pr_{y \leftarrow U_n}[h(y) \neq c(y)]. \end{aligned} \tag{57}$$

□

Given this, the proof of Theorem 3 is then as follows:

Proof (Theorem 3). We consider some concept class $\mathcal{C} \subseteq \mathcal{F}_n$ and begin by noting that for all $c \in \mathcal{C}$

$$\mathsf{Query}(\mathrm{PEX}(c, U_n)) = (x, c(x)) \text{ with } x \leftarrow U_n, \tag{58}$$

$$\mathsf{Query}(\mathrm{SAMPLE}(D_c)) = x||c(x) \text{ with } x \leftarrow U_n. \tag{59}$$

As such it is clear that any algorithm given oracle access to $\mathrm{PEX}(c, U)$ can efficiently simulate oracle access to $\mathrm{SAMPLE}(D_c)$, and vice versa. Now, we assume that $\mathcal{C}$ is efficiently classically (quantum) learnable with respect to the uniform distribution and the PEX oracle – i.e. for all valid ϵ and δ there exists an efficient classical (quantum) $(\epsilon, \delta, \mathrm{PEX}, U)$ PAC learner for $\mathcal{C}$, which we denote $\mathcal{A}_{\epsilon,\delta}$. Using this we show that for all valid ϵ, δ there exists an efficient classical (quantum) $(\epsilon, \delta, \mathrm{SAMPLE}, \mathrm{TV})$ PAC generator learner for the distribution class $\mathcal{D}_{\mathcal{C}}$, which we denote $\mathcal{A}'_{\epsilon,\delta}$. More specifically, given some valid ϵ, δ, when given access to $\mathrm{SAMPLE}(D_c)$ algorithm $\mathcal{A}'_{\epsilon,\delta}$ does the following:

1. Simulate $\mathcal{A}_{\epsilon,\delta}$ on input and obtain some $h \in \mathcal{F}_n$.
2. Output GEN_{D_h}.

By Lemma 3 we know that if $\Pr_{x \leftarrow U_n}[h(x) \neq c(x)] \leq \epsilon$, then GEN_{D_h} is a $(d_{\mathrm{TV}}, \epsilon)$ generator for D_c. Therefore it follows from the fact that $\mathcal{A}_{\epsilon,\delta}$ is an $(\epsilon, \delta, \mathrm{PEX}, U)$ efficient PAC learner for $\mathcal{C}$, that $\mathcal{A}'$ is an $(\epsilon, \delta, \mathrm{SAMPLE}, d_{\mathrm{TV}})$ PAC generator-learner for the distribution class $\mathcal{D}_{\mathcal{C}}$. □

Before continuing we note that the proof of Theorem 3 relies strongly on the fact that the concept class $\mathcal{C}$ is learnable from random examples drawn from the uniform distribution. In particular, if the concept class $\mathcal{C}$ was only learnable with respect to membership queries, or random examples drawn from some other distribution, then the distribution class learner $\mathcal{A}'$ could *not* simulate the concept class learner $\mathcal{A}$. It is this observation that motivates our restriction to the uniform distribution specific learnability of concept classes from random examples.

Given the above result, we now move onto a discussion of Conjecture 2. As per the previous discussion, if Conjecture 2 is true, this would allow one to use any concept class which is not efficiently classically PAC learnable, *with respect to the uniform distribution* and random examples (such as those discussed by Kharitonov [45, 46]), to obtain a distribution class which is not efficiently PAC generator-learnable with respect to the SAMPLE oracle and the total variation distance. As we will see, a primary obstacle in trying to proving Conjecture 2 is the non-uniqueness of exact classical generators for a given discrete probability distribution. In fact, this difficulty illustrates clearly a key difference between the learnability of Boolean functions and the generator-learnability of distribution concept classes. More specifically, given a concept class $\mathcal{C}$ which (up to some assumption) is

provably not efficiently classically learnable (with respect to random examples drawn from the uniform distribution) a natural proof strategy for Conjecture 2 would be to obtain a contradiction by showing that if the distribution concept class $\mathcal{D}_{\mathcal{C}}$ was efficiently generator-learnable, then the concept class $\mathcal{C}$ would be efficiently learnable. Similarly to the proof of Theorem 3, when given access to $\mathrm{PEX}(U, c)$ for some $c \in \mathcal{C}$, a function learner $\mathcal{A}$ for $\mathcal{C}$ could easily simulate a distribution-class learner $\mathcal{A}'$ for $\mathcal{D}_{\mathcal{C}}$ by using the $\mathrm{PEX}(U, c)$ oracle to simulate the $\mathrm{SAMPLE}(D_c)$ oracle. However, unlike in the proof of Theorem 3, the concept class learner $\mathcal{A}$ cannot simply extract a function hypothesis h from the approximate generator output by $\mathcal{A}'$. To make this more precise, and to pinpoint clearly the key difficulty, we begin with the following series of observations and lemmas which fully characterize the non-uniqueness of *exact* classical generators for D_c.

Observation 4. *For all $m \geq n$ the generator $\mathsf{GEN}_{(D_c,m)} : \{0,1\}^m \to \{0,1\}^{n+1}$ defined via*

$$\mathsf{GEN}_{(D_c,m)}(x) := \mathsf{GEN}_{D_c}(x_{[1,n]}), \tag{60}$$

is an exact generator for D_c, where $x := x_{[1,n]}||x_{[n+1,m]}$.

Proof. Let D be the distribution generated by $\mathsf{GEN}_{(D_C,m)}$. Then, for all $y \in \{0,1\}^n$ we have that

$$\begin{aligned} D(y||c(y)) &= \frac{1}{2^m} \sum_{x \in \{0,1\}^m} \delta(\mathsf{GEN}_{(D_c,m)}(x), y||c(y)) \\ &= \frac{1}{2^m} \sum_{x \in \{0,1\}^m} \delta(\mathsf{GEN}_{D_c}(x_{[1,n]}), \mathsf{GEN}_{D_c}(y)) \\ &= \frac{1}{2^m} \sum_{x \in \{0,1\}^m} \delta(x_{[1,n]}, y) \\ &= \frac{1}{2^m} 2^{m-n} \\ &= \frac{1}{2^n}. \end{aligned} \tag{61}$$

□

Observation 5. *For all $m \geq n$, and for all permutations $P : \{0,1\}^m \to \{0,1\}^m$ the generator $\mathsf{GEN}_{(D_c,m,P)} : \{0,1\}^m \to \{0,1\}^{n+1}$, defined via*

$$\mathsf{GEN}_{(D_c,m,P)}(x) := \mathsf{GEN}_{(D_c,m)}(P(x)), \tag{62}$$

is an exact generator for D_c.

Proof. As P is a permutation it maps uniformly random inputs to uniformly random outputs, and as such the distribution over output strings of the generator is unaffected by composition with a permutation. □

Next, we note that we can rule out the possibility of an exact generator with $m < n$:

Lemma 4. *Let $\mathsf{GEN}_D : \{0,1\}^m \to \{0,1\}^{n+1}$ be an exact classical generator for some distribution $D \in \mathcal{D}_{n+1}$. Then*

$$d_{\mathrm{TV}}(D, D_c) \geq 1 - 2^{m-n}. \tag{63}$$

Proof. We recall that if $\mathsf{GEN}_D : \{0,1\}^m \to \{0,1\}^{n+1}$ then for all $y \in \{0,1\}^{n+1}$

$$D(y) = \frac{1}{2^m} \sum_{x \in \{0,1\}^m} \delta(\mathsf{GEN}_D(x), y). \tag{64}$$

From the above it follows that there exist at most 2^m strings $y \in \{0,1\}^{n+1}$ with non-zero probability, and all such strings have probability $\alpha/2^m$ for some $\alpha \in \{1, \ldots, 2^m\}$. Next, we note that

$$D_c(y) = \begin{cases} \frac{1}{2^n} \text{ if } y = x||c(x) \text{ for some } x \in \{0,1\}^n, \\ 0 \text{ otherwise.} \end{cases} \tag{65}$$

As such, we see that for $m \leq n$, the optimal $\mathsf{GEN}_D : \{0,1\}^m \to \{0,1\}^{n+1}$ which minimizes $d_{\mathrm{TV}}(D, D_c)$ is the one which assigns a probability of $1/2^m$ to 2^m of the 2^n strings $y \in \{0,1\}^{n+1}$ which are of the form $x||c(x)$ for some $x \in \{0,1\}^n$, and a zero probability to the remaining $2^n - 2^m$ strings. For this optimal generator, we have that

$$\begin{aligned} d_{\mathrm{TV}}(D, D_c) &= \frac{1}{2} \sum_{y \in \{0,1\}^{n+1}} |D(y) - D_c(y)| \\ &= \frac{1}{2} \sum_{x \in \{0,1\}^n} |D(x||c(x)) - \frac{1}{2^n}| \\ &= \frac{1}{2} \left(2^m \left(\frac{1}{2^m} - \frac{1}{2^n} \right) + (2^n - 2^m) \frac{1}{2^n} \right) \\ &= 1 - 2^{m-n}. \end{aligned} \tag{66}$$

The statement follows from the fact that the above was calculated for the optimal generator. □

An immediate corollary of Lemma 4, ruling out the possibility of an exact generator for D_c with $m < n$, is then as follows:

Corollary 3.1. *Given some $\mathsf{GEN}_D : \{0,1\}^m \to \{0,1\}^{n+1}$, if $d_{\mathrm{TV}}(D, D_c) < 1/2$, then $m \geq n$.*

Finally, given the above results, the following lemma allows us to fully characterize all possible classical exact generators for D_c.

Lemma 5. *$\mathsf{GEN}_D : \{0,1\}^m \to \{0,1\}^{n+1}$ is an exact classical generator for D_c if and only if $m \geq n$ and $\mathsf{GEN}_D = \mathsf{GEN}_{(D_c,m,P)}$ for some permutation $P : \{0,1\}^m \to \{0,1\}^m$.*

Proof. The one direction of the above statement is precisely the content of observations 4 and 5. In the other direction, the fact that $m \geq n$ follows directly from Corollary 3.1. Then, in order for GEN_D to be an exact classical generator for D_c, we know that for all $y \in \{0,1\}^n$ there are exactly 2^{m-n} strings $x \in \{0,1\}^m$ for which

$$\mathsf{GEN}_D(x) = y||c(y) = \mathsf{GEN}_{D_c}(y). \tag{67}$$

Let us denote the set of these strings as

$$X_y = \{x \in \{0,1\}^m \,|\, \mathsf{GEN}_D(x) = \mathsf{GEN}_{D_c}(y)\} \tag{68}$$

Additionally, let us denote by $\tilde{X}_y$ the set of all strings $x \in \{0,1\}^m$ for which $\mathsf{GEN}_{(D_c,m)}(x) = \mathsf{GEN}_{D_c}(y)$, i.e.

$$\begin{aligned} \tilde{X}_y &= \{x \in \{0,1\}^m \,|\, \mathsf{GEN}_{(D_c,m)}(x) = \mathsf{GEN}_{D_c}(y)\} \\ &= \{x \in \{0,1\}^m \,|\, x_{[1,n]} = y\}. \end{aligned} \tag{69}$$

Now, as $|X_y| = |\tilde{X}_y| = 2^{m-n}$ we can define a permutation $P_y : X_y \to \tilde{X}_y$. Additionally, note that for all $x \in \{0,1\}^m$, there is exactly one $y \in \{0,1\}^n$ such that $x \in X_y$, and given this, we can define a permutation $P : \{0,1\}^m \to \{0,1\}^m$ via

$$P(x) = P_y(x) \text{ if } x \in X_y. \tag{70}$$

Using this, we have that

$$\mathsf{GEN}_{(D_c,m,P)}(x) = \mathsf{GEN}_{(D_c,m)}(P(x)) = \mathsf{GEN}_D(x) \tag{71}$$

for all $x \in \{0,1\}^m$. □

Given the above characterization, let us return to a discussion of one natural strategy to prove Conjecture 2, and the fundamental obstacle one faces with this strategy. As mentioned before, given a concept class $\mathcal{C}$ which is provably hard to learn, a natural strategy would be to assume that the distribution concept class $\mathcal{D}_\mathcal{C}$ *is* efficiently PAC generator-learnable, and use this assumption to show that the concept class $\mathcal{C}$ is efficiently PAC learnable, which would be a contradiction. To simplify the exposition, let us make the stronger assumption that the concept class $\mathcal{D}_\mathcal{C}$ is *exactly* learnable – i.e. for all δ there exists a generator learner $\mathcal{A}'_\delta$ which when given access to $\mathrm{SAMPLE}(D_c)$ outputs with probability $1 - \delta$ an *exact* generator GEN_D for D_c. Given this assumption, we want to construct an efficient $(\epsilon, \delta, \mathrm{PEX}, U)$ learner $\mathcal{A}_{\epsilon,\delta}$ for the concept class $\mathcal{C}$. A natural approach would be as follows:

1. When given access to $\mathrm{PEX}(U, c)$, the learner $\mathcal{A}_{\epsilon,\delta}$ simulates $\mathcal{A}'_\delta$ and obtains some generator GEN_D, which with probability $1 - \delta$ is an exact generator for D_c.

2. Using GEN_D, the learner $\mathcal{A}'_{\epsilon,\delta}$ outputs some hypothesis $h \in \mathcal{F}_n$.

Now, as per Lemma 5, we know that if GEN_D is an exact generator for D_c, then $\mathsf{GEN}_D = \mathsf{GEN}_{(D_c,m,P)}$ for some $m \geq n$ and some permutation $P : \{0,1\}^m \to \{0,1\}^m$. Using this information, and given an exact generator for D_c, how should the learner construct its output hypothesis h? Well, note that for all $y \in \{0,1\}^m$ for which $y_{[1,n]} = x$ we have that

$$\begin{aligned}
\left[\mathsf{GEN}_{(D_c,m,P)}(P^{-1}(y))\right]_{[n+1]} &= \left[\mathsf{GEN}_{(D_c,m)}(P(P^{-1}(y)))\right]_{[n+1]} \\
&= \left[\mathsf{GEN}_{(D_c,m)}(y)\right]_{[n+1]} \\
&= \left[\mathsf{GEN}_{(D_c)}(y_{[1,n]})\right]_{[n+1]} \\
&= \left[\mathsf{GEN}_{(D_c)}(x)\right]_{[n+1]} \\
&= [x||c(x)]_{[n+1]} \\
&= c(x). \qquad (72)
\end{aligned}$$

As such, if the learner $\mathcal{A}$ knew or could learn the permutation P^{-1}, then it could simply output the hypothesis $h \in \mathcal{F}_n$ defined via

$$h(x) = \left[\mathsf{GEN}_{(D_c,m,P)}(P^{-1}(x||0\ldots0))\right]_{[n+1]} = c(x) \qquad (73)$$

which would in fact be an exact solution. The key point, however, is that the learner $\mathcal{A}$ *does not* even know the permutation P. A natural question is then whether $\mathcal{A}$ could use $\mathsf{GEN}_{(D_c,m,P)}$ to *learn* P^{-1}? Well, we note that

$$\mathsf{GEN}_{(D_c,m,P)}(x) = P(x)_{[1,n]}||c(P(x)_{[1,n]}) \qquad (74)$$

and so, at least in the case that $m = n$, it is clear that one can generate a data-set of input/output pairs $(x, P(x)) := (P^{-1}(y), y)$. Unfortunately, however, it is known that even with respect to membership queries, there does not exist an efficient *exact* learner for the concept class of permutations [47], and so the possibility of efficiently *exactly* learning P^{-1} from $\mathsf{GEN}_{(D_c,m,P)}$ is ruled out. Of course, in principle it could be sufficient to learn an approximation to P^{-1} from polynomially many random examples, however whether or not this is possible efficiently is not known. Additionally, as mentioned before, all of this is under the overly strong assumption that the generator learner is an exact learner, which outputs an exact generator with $m = n$. As can be seen from the above discussion, lifting either of these assumptions makes the fundamental problem of defining a suitable hypothesis from the output generator significantly harder.

At this stage we have outlined the primary difficulty with one natural strategy for proving Conjecture 2, which provides a clear illustration of a key conceptual and technical difference between the PAC learnability of Boolean function concept classes and the generator learnability of distribution concept classes. Of course, one could conceive of a variety of other strategies, based for example on alternative characterizations of efficient PAC learnabilty [48], Occam algorithms [49] or VC dimensions [50], however it is important to keep in mind the restriction of *efficient learnability with respect to random examples from the uniform distribution*, which makes it unclear how to immediately apply existing results involving some of the above mentioned tools and characterizations.

5. DISCUSSION AND CONCLUSION

Given the results and insights of this work, we provide here a brief summary, as well as a perspective on interesting open questions and future directions. Firstly, to summarize, in Section 3 we have constructed a class of probability distributions, specified by *classical* generators, which under the DDH assumption, is provably not efficiently PAC generator-learnable by any classical algorithm, but for which we have constructed an efficient quantum generator-learner, which only requires access to *classical* samples from the unknown distribution. This construction therefore provides a clear example of a generative modelling task for which quantum learners exhibit a provable quantum advantage. Despite this, there of course remain a variety of interesting open questions, for which the insights and conjectures from Section 4 may provide useful:

1. What can one say about the quantum versus classical PAC learnability (in a generative sense) of the probability distributions used for the demonstration of "quantum computational supremacy" [23–25] – i.e. probability distributions which can be efficiently generated by quantum processes, but for which there are no efficient classical generators. In particular, there are two distinct questions: Firstly, while it is known that one cannot efficiently sample, even approximately, from such distributions using purely classical means, is it possible to prove that when *given* samples from such distributions it is also not possible to efficiently classically learn a description of the underlying efficient quantum generator? Intuitively, this question seems closely related to the question of whether or not efficient classical *verification* of such distributions is

possible. To this end, we note that Ref. [51] has proven that efficiently verifying certain such distributions given classical samples is not efficiently possible. Interestingly, this impossibility of efficient black-box verification is closely related to the hardness of sampling. Conversely, it seems plausible that the existence of an efficient classical PAC generator-learner would imply the existence of an efficient classical black-box verification algorithm – which would then rule out the possibility of an efficient classical PAC generator-learner. However, as discussed at length in Section 4.2, it is important to note that there is no *unique* generator for a given probability distribution, and as such, while plausible, it is not clear how exactly to exploit an efficient PAC-generator learner for the construction of an efficient black-box verification algorithm, and formalizing this connection would certainly be of great interest. Secondly, in addition to proving hardness of classically learning such distributions, we would of course also like to investigate the possibility of efficient quantum learnability, and how this relates to quantum verifiability [51–55]. Once again, while it is known that there exist efficient quantum generators for such distributions, this certainly does not immediately imply the existence of efficient quantum PAC generator-learners. As such, understanding whether or not there exist efficient quantum PAC generator-learners for such distributions is of natural interest, particularly in light of the potential connections between generator-learnability and black-box verification.

2. As the quantum learner that we have constructed in Section 3.3 relies on the quantum algorithm for discrete logarithms [28], it is most likely the case that the quantum advantage exhibited by this learner would require the existence of a universal fault-tolerant quantum computer. Given the current availability of NISQ devices, it is of natural interest to ask whether there exists a generative modelling problem for which near-term quantum learners can exhibit a provable advantage over classical learning algorithms. In order to answer this question it will certainly be necessary to understand better the theoretical properties of previously proposed NISQ hybrid-quantum classical algorithms for generative modelling, such as Born machines [17]. Additionally, it also seems likely that techniques for proving classical hardness results under weaker assumptions, as discussed in Section 4, would be of great help. Alternatively, it may help to focus on probability distributions which can be generated by near term quantum devices, but not classical devices, as discussed in the previous point. It is also important to reiterate that the seperation we have obtained here relies fundamentally on the known advantage quantum computers offer for computing discrete logarithms, and as such this work does not provide a new primitive for proving classical/quantum separations. Whether one can construct a quantum/classical learning separation without relying on such prior primitives is an interesting open question.

3. It is of interest to note that the efficient quantum generator-learner that we have constructed in Section 3.3 requires only a *single* oracle query, and *always* outputs an *exact* generator. Such a quantum generator-learner is certainly formally sufficient for the purpose of answering Question 1 in the affirmative, and from one perspective provides the "optimal" generator-learner, in the sense that its query complexity is clearly optimal, and its behaviour (both run-time and output) is independent of ϵ and δ – i.e. for all ϵ and δ the algorithm returns an exact generator with certainty. However, intuitively we might think of a "learning" algorithm as an algorithm which requires multiple samples (i.e. learns from a "data-set"), and outputs most often only approximate solutions, and from this perspective it is not clear to which extent the generator-learner we have constructed can be considered a "learning algorithm". As such, from a conceptual perspective it is interesting to ask whether there exists a distribution concept class which provides an affirmative answer to Question 1, but for which the efficient quantum generator learner requires a non-trivial query complexity, and at best outputs only a suitably approximate generator, with sufficiently high probability. In particular, while the generator-learner we have constructed is clearly highly specific to the distribution concept class we have constructed, it is possible that by considering concept classes for which always exact constant query complexity learners do not exist, one may be forced to construct or consider quantum-generative modelling algorithms which are not as task-specific as the learner we have constructed here, and may also be suitable for near-term devices.

4. In this work we have considered quantum and classical generator-learners, both of which only have access to *classical* samples from the unknown probability distribution – i.e. to the SAMPLE oracle. Analogously to the Boolean function setting [4], it is also interesting to ask whether there exists a distribution concept class for which a quantum learner exhibits a quantum advantage, but only if the quantum learner has access to *quantum* samples from the $\mathrm{QSAMPLE}$ oracle. As discussed in Section 4.1, it seems likely that if Conjecture 1 is true then one could construct such a concept class using the weak-secure pseudorandom function collection based on the Learning Parity with Noise problem. Additionally, it is also plausible that if Conjecture 2 is true, then one could modify both this and Theorem 3 to prove both learnability and hardness results for quantum learners with quantum samples, from corresponding results for Boolean function concept classes.

ACKNOWLEDGMENTS

RS is funded by the BMWi, under the PlanQK initiative, and is grateful for many fruitful discussions with Alexander Nietner, Nathan Walk and the QML reading group at the FU Berlin. JPS acknowledges funding of the Berlin Institute for the Foundations of Learning and Data, the Einstein Foundation Berlin and the BMBF "Post-Quantum-Cryptography" framework. JE is funded by

the BMWi under the PlanQK initiative and the DFG (CRC 183, Project B01, and EI 519/14-1). DH is funded by the Templeton Foundation and is grateful for discussions with Brian Coyle.

[1] Leslie G. Valiant, "A theory of the learnable," Communications of the ACM **27**, 1134–1142 (1984).
[2] Michael J. Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani, *An introduction to computational learning theory* (MIT press, 1994).
[3] Shai Shalev-Shwartz and Shai Ben-David, *Understanding machine learning: From theory to algorithms* (Cambridge University Press, 2014).
[4] Srinivasan Arunachalam and Ronald de Wolf, "Guest column: A survey of quantum learning theory," ACM SIGACT News **48**, 41–67 (2017).
[5] Carlo Ciliberto, Andrea Rocchetto, Alessandro Rudi, and Leonard Wossnig, "The statistical limits of supervised quantum learning," (2020), arXiv preprint arXiv:2001.10477.
[6] Vedran Dunjko and Hans J Briegel, "Machine learning & artificial intelligence in the quantum domain: a review of recent progress," Rep. Prog. Phys. **81**, 074001 (2018).
[7] Maria Schuld and Francesco Petruccione, *Supervised learning with quantum computers* (Springer, 2018).
[8] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd, "Quantum machine learning," Nature **549**, 195–202 (2017).
[9] Nader H. Bshouty and Jeffrey C. Jackson, "Learning DNF over the uniform distribution using a quantum example oracle," SIAM Journal on Computing **28**, 1136–1153 (1998).
[10] Clément L. Canonne, "A short note on learning discrete distributions," arXiv preprint arXiv:2002.11457 (2020).
[11] Sudeep Kamath, Alon Orlitsky, Dheeraj Pichapati, and Ananda Theertha Suresh, "On learning distributions from their samples," in *Conference on Learning Theory* (2015) pp. 1066–1100.
[12] Ilias Diakonikolas, "Learning structured distributions." Handbook of Big Data **267** (2016).
[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Advances in neural information processing systems* (2014) pp. 2672–2680.
[14] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," arXiv preprint arXiv:1312.6114 (2013).
[15] Ivan Kobyzev, Simon Prince, and Marcus Brubaker, "Normalizing flows: An introduction and review of current methods," IEEE Transactions on Pattern Analysis and Machine Intelligence , doi: 10.1109/TPAMI.2020.2992934 (2020).
[16] Jin-Guo Liu and Lei Wang, "Differentiable learning of quantum circuit Born machines," Phys. Rev. A **98**, 062324 (2018).
[17] Brian Coyle, Daniel Mills, Vincent Danos, and Elham Kashefi, "The Born supremacy: quantum advantage and training of an Ising Born machine," npj Quantum Information **6**, 60 (2020).
[18] Pierre-Luc Dallaire-Demers and Nathan Killoran, "Quantum generative adversarial networks," Phys. Rev. A **98** (2018), 10.1103/physreva.98.012324.
[19] Ling Hu, Shu-Hao Wu, Weizhou Cai, Yuwei Ma, Xianghao Mu, Yuan Xu, Haiyan Wang, Yipu Song, Dong-Ling Deng, Chang-Ling Zou, and Luyan Sun, "Quantum generative adversarial learning in a superconducting quantum circuit," Science Advances **5**, eaav2761 (2019).
[20] Seth Lloyd and Christian Weedbrook, "Quantum generative adversarial learning," Phys. Rev. Lett. **121**, 040502 (2018).
[21] Shouvanik Chakrabarti, Huang Yiming, Tongyang Li, Soheil Feizi, and Xiaodi Wu, "Quantum Wasserstein generative adversarial networks," in *Advances in Neural Information Processing Systems* (2019) pp. 6781–6792.
[22] Guillaume Verdon, Jacob Marks, Sasha Nanda, Stefan Leichenauer, and Jack Hidary, "Quantum Hamiltonian-ased models and the variational quantum thermalizer algorithm," (2019), arXiv preprint arXiv:1910.02071.
[23] Scott Aaronson and Alexander Arkhipov, "The computational complexity of linear optics," (2011), arXiv:1011.3245.
[24] Michael J. Bremner, Ashley Montanaro, and Daniel J. Shepherd, "Average-case complexity versus approximate simulation of commuting quantum computations," Phys. Rev. Lett. **117**, 080501 (2016).
[25] 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, *et al.*, "Quantum supremacy using a programmable superconducting processor," Nature **574**, 505–510 (2019).
[26] Dan Boneh, "The decision diffie-hellman problem," in *International Algorithmic Number Theory Symposium* (Springer, 1998) pp. 48–63.
[27] Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie, "On the learnability of discrete distributions," in *Proceedings of the Twenty-sixth Annual ACM Symposium on Theory of Computing*, STOC '94 (ACM, New York, NY, USA, 1994) pp. 273–282.
[28] Michele Mosca and Christof Zalka, "Exact quantum fourier transforms and discrete logarithm algorithms," International Journal of Quantum Information **2**, 91–100 (2004).
[29] Oded Goldreich, Shafi Goldwasser, and Silvio Micali, "How to construct random functions," J. ACM **33**, 792–807 (1986).
[30] Erich L. Lehmann and Joseph P. Romano, *Testing statistical hypotheses* (Springer Science & Business Media, 2006).
[31] Oded Goldreich, *Foundations of cryptography: volume 1, basic tools* (Cambridge University Press, 2007).
[32] Mark Zhandry, "How to construct quantum random functions," in *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12 (IEEE Computer Society, Washington, DC, USA, 2012) pp. 679–687.
[33] Andrej Bogdanov and Alon Rosen, "Pseudorandom functions: Three decades later," in *Tutorials on the Foundations of Cryptography* (Springer, 2017) pp. 79–158.
[34] Jonathan Katz and Yehuda Lindell, *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)* (Chapman & Hall/CRC, 2007).
[35] Manuel Blum and Silvio Micali, "How to generate cryptographically strong sequences of pseudorandom bits," SIAM journal on Computing

13, 850–864 (1984).
[36] Moni Naor and Omer Reingold, "Number-theoretic constructions of efficient pseudo-random functions," Journal of the ACM (JACM) **51**, 231–262 (2004).
[37] Reza Rezaeian Farashahi, Berry Schoenmakers, and Andrey Sidorenko, "Efficient pseudorandom generators based on the ddh assumption," in *International Workshop on Public Key Cryptography* (Springer, 2007) pp. 426–441.
[38] Krzysztof Pietrzak, "Cryptography from learning parity with noise," in *International Conference on Current Trends in Theory and Practice of Computer Science* (Springer, 2012) pp. 99–114.
[39] Andrew W. Cross, Graeme Smith, and John A. Smolin, "Quantum learning robust against noise," Phys. Rev. A **92**, 012327 (2015).
[40] Alex B. Grilo, Iordanis Kerenidis, and Timo Zijlstra, "Learning-with-errors problem is easy with quantum samples," Phys. Rev. A **99**, 032314 (2019).
[41] Diego Riste, Marcus P. da Silva, Colm A. Ryan, Andrew W. Cross, Antonio D. Córcoles, John A. Smolin, Jay M. Gambetta, Jerry M. Chow, and Blake R. Johnson, "Demonstration of quantum advantage in machine learning," npj Quantum Inf. **3**, 1–5 (2017).
[42] Michael Kearns and Leslie Valiant, "Cryptographic limitations on learning boolean formulae and finite automata," Journal of the ACM (JACM) **41**, 67–95 (1994).
[43] Rocco A. Servedio and Steven J. Gortler, "Equivalences and separations between quantum and classical learnability," SIAM J. Comp. **33**, 1067–1092 (2004).
[44] Karsten Verbeurgt, "Learning DNF under the uniform distribution in quasi-polynomial time," in *Proceedings of the third annual workshop on Computational learning theory* (1990) pp. 314–326.
[45] Michael Kharitonov, "Cryptographic hardness of distribution-specific learning," in *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing* (1993) pp. 372–381.
[46] Michael Kharitonov, "Cryptographic lower bounds for learnability of boolean functions on the uniform distribution," Journal of Computer and System Sciences **50**, 600–610 (1995).
[47] Vikraman Arvind and N. Variyam Vinodchandran, "The complexity of exactly learning algebraic concepts," in *International Workshop on Algorithmic Learning Theory* (Springer, 1996) pp. 100–112.
[48] David Haussler, Michael Kearns, Nick Littlestone, and Manfred K. Warmuth, "Equivalence of models for polynomial learnability," Information and Computation **95**, 129–161 (1991).
[49] Raymond Board and Leonard Pitt, "On the necessity of occam algorithms," Theoretical Computer Science **100**, 157–184 (1992).
[50] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," Journal of the ACM (JACM) **36**, 929–965 (1989).
[51] Dominik Hangleiter, Martin Kliesch, Jens Eisert, and Christian Gogolin, "Sample complexity of device-independently certified quantum supremacy," Phys. Rev. Lett. **122**, 210502 (2019).
[52] Xun Gao, Sheng-Tao Wang, and Lu-Min Duan, "Quantum supremacy for simulating a translation-invariant Ising spin model," Phys. Rev. Lett. **118**, 040502 (2017).
[53] Juani Bermejo-Vega, Dominik Hangleiter, Martin Schwarz, Robert Raussendorf, and Jens Eisert, "Architectures for quantum simulation showing a quantum speedup," Phys. Rev. X **8**, 021010 (2018).
[54] Jacob Miller, Stephen Sanders, and Akimasa Miyake, "Quantum supremacy in constant-time measurement-based computation: A unified architecture for sampling and verification," Phys. Rev. A **96**, 062320 (2017), arXiv:1703.11002.
[55] Masahito Hayashi and Yuki Takeuchi, "Verifying commuting quantum computations via fidelity estimation of weighted graph states," New J. Phys. **21**, 093060 (2019), 1902.03369.