Publicly-Verifiable Certificates for Statistical Algorithms

Michael Ngo*
MIT
mingo@mit.edu

Michael P. Kim†
Cornell University
mpk@cs.cornell.edu

1 April 2026

Abstract

Following Goldwasser, Rothblum, Shafer, and Yehudayoff, who defined a framework for interactive proofs of learning [GRSY21], we initiate the study of non-interactive proofs of learning. We define and study a new notion: *Publicly-Verifiable Certificates of Statistical Validity* (pvCSVs), which allow for public, distributionally-robust certification that the result of a learning algorithm is valid. In a pvCSV, a learner publishes a hypothesis h and corresponding certificate π; then, *any* user, who holds a user-specific distribution, can read the pair (h, π) and determine efficiently whether the hypothesis is valid *according to the user-specific distribution*.

We construct pvCSVs in the context of Adaptive Statistical Query (SQ) Algorithms. To certify SQ algorithms that makes k adaptive queries, we construct pvCSVs where the sample complexity scales with $O(\log k)$, whereas the sample complexity of the best learning algorithms scale with $\tilde{O}(\sqrt{k})$. More generally, we study proof systems for learning in the SQ model, demonstrating the model's strengths as well as its limitations.

1 Introduction

Training large-scale AI models using statistical machine learning is notoriously costly. Due to the resource demands of running ML algorithms, AI users rely upon pre-trained models from a handful of tech companies. These companies hold enough data—so they claim—to train general-purpose models that are effective across a wide range of settings. In this setup, however, users receive no guarantee that the AI models were trained appropriately. If users fear that the training data does not adequately reflect their setting, they must investigate, for themselves, whether the model produces errors (or worse, harms) within their application.

Inspired by these issues, Goldwasser, Shafer, Rothblum, and Yehudayoff [GRSY21] introduced and studied the problem of delegating machine learning, through the PAC Verification framework. Building on the classic model of interactive proofs [GMR85,Bab85], PAC Verification formalizes the problem: a statistically-limited user (the *verifier*) interacts with a powerful, but untrusted learner

*Research completed while at Cornell University, supported by the Bowers Undergraduate Research Experience (BURE) and the Dean Archer Undergraduate Research Program. This research was supported by a gift from Google.
†This research was supported by a gift from Google.

(the *prover*), who aims to convince the user that a given model (the *hypothesis*) is valid. As in cryptographic proof systems, PAC Verification requires that protocols for delegating learning satisfy formal notions of *completeness* and *soundness*. To date, results in the area focus on developing protocols for verifying Agnostic PAC Learning [Val84, Hau92, KSS94] for specific concept classes [GRSY21, MS23, GJK+24]. As a notable exception, Mutreja and Shafer [MS23] also introduce a notion of delegation of Statistical Query (SQ) algorithms for loss minimization.

An essential element of prior works on delegation of learning is *interaction*. To establish whether a given hypothesis is valid, the prover and verifier exchange a sequence of messages in an online fashion, after which the verifier chooses to accept or reject the prover's hypothesis. As a concrete example, the original work on PAC verification shows how to delegate the Goldreich-Levin algorithm [GL89] (which requires point query access to the unknown function), when the verifier only has i.i.d. labeled samples. In this proof system, the verifier uses its interactions with the prover to label the point queries, while cleverly hiding some points whose labels are known to the verifier to ensure soundness. The early works on PAC verification have demonstrated that interactive proof systems provide a powerful tool for efficiently checking the results of an expensive ML computation.[1]

Interaction, however, also presents challenges. Most immediately, running an interactive proof requires the verifier and prover to be online at the same time to execute the protocol. Additionally, each execution of the interactive proof may require the prover to answer execution-specific challenges, including re-running the original ML computation. Given the immense cost of training ML models in the first place, providers may be unwilling to participate in the interactive proof more than once (if at all). In such a setting, where the interactive proof is executed once—between the learner and a single verifier—many users would have to place their trust in a single entity. Even if users agree that the verifier is generally trustworthy, as before, individual users may worry that the verifier's data does not represent their setting and applications.

This Work. We initiate the study of non-interactive proofs of learning. Our study leads us to a new notion: *Publicly-Verifiable Certificates of Statistical Validity (pvCSVs)*. pvCSVs allow for public, distributionally-robust certification that the result of a learning algorithm is valid. In particular, a pvCSV allows a learner to publish a hypothesis h and certificate of validity π that allows *any* downstream user to subsequently verify that the hypothesis is statistically valid on a *user-specified* distribution.

We can understand the semantics of a pvCSV by imagining two worlds.

- In the first world, a user collects a huge amount of data from a distribution $\mathcal{D}$, relevant to their setting and application. Then, they correctly execute a statistical learning algorithm $\mathcal{A}$ on top of this data to obtain a hypothesis h_{ideal}.
- In the second world, a centralized, well-resourced learner publishes a pvCSV (h_{real}, π) for the algorithm $\mathcal{A}$; the same user from before collects a much more modest amount of data from $\mathcal{D}$, then reads and verifies (h_{real}, π), using the small amount of user-specific data.

A pvCSV guarantees that if (h_{real}, π) passes verification, the two worlds produced equally-valid hypotheses $h_{\text{ideal}} \approx_{\mathcal{D},\mathcal{A}} h_{\text{real}}$ according to the user-specific data distribution and the learning algo-

[1] In these works, and in our paper, "efficiency" primarily focuses on statistical resources, rather than computation.

rithm $\mathcal{A}$. This guarantee must hold even though the user has no knowledge of the data distribution used to produce the pvCSV. As such, pvCSVs resolve the key shortcomings of interactive proofs for delegation of learning: the learner can execute the expensive training algorithm once and (with little overhead) generate a corresponding pvCSV certificate that any user can subsequently verify.

Certification of Adaptive Data Analysis. We make our study of pvCSVs concrete by revisiting the question of Adaptive Data Analysis [DFH+15c] in the Statistical Query Model [Kea98]. Many tools for learning from data—including workhorse ML algorithms like gradient descent—can be framed as adaptive statistical algorithms. In such an algorithm, the learner is allowed to ask a sequence of queries of the data distribution (e.g., *What is the gradient of the expected loss over $\mathcal{D}$?*), where each query may depend on the results from prior queries.

Formally, we consider learning algorithms $\mathcal{A}$ that interact with a Statistical Query (SQ) oracle $\mathcal{O}$: given a tolerance τ and a query q, $\mathcal{O}(q)$ responds with a τ-accurate estimate of the expectation of the predicate q over the data distribution. Critical to our investigation, the algorithm may select its sequence of queries *adaptively* based on the prior responses. That is, the algorithm's choice of the i-th query q_i may depend arbitrarily on prior queries $q_1, \ldots, q_{i-1}$ and responses $\mathcal{O}(q_1), \ldots, \mathcal{O}(q_{i-1})$, (e.g., *What is the gradient at the i-th iterate, after taking $i-1$ gradient descent steps?*).

While the paradigm of adaptive data analysis is a versatile and powerful tool for learning, such algorithms are known to be statistically expensive. About a decade ago, [DFH+15c] identified adaptivity as a key issue in statistical algorithms. To maintain the statistical validity of an adaptive analysis requires the learner to either resample fresh data to support every new query, or to employ sophisticated (differentially-private) algorithms for answering queries in a way to prevent overfitting to the data set [DFH+15c, DFH+15a, DFH+15b, BNS+16, FS18, JLN+19, DK22, Bla25]. To answer k adaptively selected statistical queries, the best algorithms use a number of samples scaling (roughly) with $\sqrt{k}$, and in fact, this dependence is essentially tight [HU14, SU15].

In other words, no matter what techniques are employed, adaptive statistical algorithms require *exponentially* more data than non-adaptive (batch) statistical analyses of similar size. In this work, we ask when we can certify the results of adaptive data analysis more efficiently than learning.

1.1 Our Contributions

We develop proof systems for delegating *arbitrarily-adaptive* statistical algorithms where the verifier requires samples scaling only with the *non-adaptive* complexity. In doing so, we realize an *exponential* gap, between the sample complexity required for executing SQ algorithms versus verifying them. Moving beyond prior work on *interactive* proofs for learning, we build novel *non-interactive* proof systems—namely, *publicly-verifiable Certificates of Statistical Validity*—that enable a new form of distributionally-robust validation of statistical learning. Along the way, we develop a number of extensions to earlier models of proof systems for delegation of learning.

Publicly-Verifiable Certificates of Statistical Validity. In Section 3, we introduce our primary contribution: a new notion of proof that allows for public, distributionally-robust certification of learning. A *publicly-verifiable Certificate of Statistical Validity* (pvCSV) is a non-interactive proof

system that allows any verifier to certify the results of a statistical computation *with respect to the verifier's own distribution.* A pvCSV allows a single, well-resourced learner—the prover—to publish the results of a statistical algorithm in a way that can be checked efficiently (with less resources) by any downstream verifier; in particular, the verifiers need not hold the same distribution as the prover. Instead, the proof system guarantees a *universal soundness* property such that if the verifier accepts the proof, then the results of the statistical algorithm are valid on the verifier's distribution—even if the algorithm was executed using samples from a different distribution.

Definition 1 (pvCSV, informal)**.** *A publicly-verifiable Certificate of Statistical Validity is a non-interactive proof system, where a prover P with distribution $\mathcal{D}_P$ publishes a hypothesis h paired with certificate π. Any verifier V with distribution $\mathcal{D}_V$ can read the pair (h, π) and accept or reject where the following guarantees hold with high probability.*

- *Completeness: if $\mathcal{D}_P = \mathcal{D}_V$, there exists an honest prover pair (h, π) such that h is valid for $\mathcal{D}_V$ and V accepts.*
- *Universal Soundness: for any verifier V with distribution $\mathcal{D}_V$, for any (possibly-cheating) prover pair $(\tilde{h}, \tilde{\pi})$, if V accepts, then $\tilde{h}$ is actually valid for $\mathcal{D}_V$.*

One way to understand the guarantee of pvCSVs is as a robust proof of statistical validity, *without an explicit assumption about distributional shifts.* Rather than positing some known relationship between the prover and verifier distributions, the verification procedure works for any $\mathcal{D}_P$ and $\mathcal{D}_V$ and results in acceptance whenever the published certificate—derived from the execution of a learning algorithm $\mathcal{A}$ using the prover's distribution—reflects some legitimate execution on the verifier's distribution. While our notion of completeness assumes that $\mathcal{D}_P = \mathcal{D}_V$, the guarantee is more subtle than this equality would suggest. The distributions $\mathcal{D}_P$ and $\mathcal{D}_V$ may differ significantly in composition, but if the verifier accepts, then (by universal soundness) the hypothesis h is valid for $\mathcal{D}_V$ (because the distributions are indistinguishable according to some invocation of $\mathcal{A}$). The careful reader will note that this "universal soundness" condition is actually implied by standard soundness for delegation of learning,[2] but holds new significance in the context of non-interactive proofs. Our view on soundness, paired with a non-interactive proof system, allows us to realize certificates of learning that can be verified publicly by any user.

We can use various measures of complexity to evaluate the quality of a pvCSV construction. Principle to our work is sample complexity: we aim for pvCSVs where the verifier saves considerably in the number of samples needed from $\mathcal{D}_V$ compared to the number of samples required for learning (or proving). Further, we can track other measures like time complexity (of both the verifier and the honest prover) and proof length as well. We define pvCSVs formally in Section 3 and provide a thorough discussion of the notion and its properties (like validity and universal soundness) therein.

With this key definition in place, the main technical contributions of this work are to construct pvCSVs for adaptive statistical algorithms within the SQ learning framework. Our pvCSVs achieve an *exponential* gap in the sample complexity between SQ verification and SQ learning. While the core idea of each of our constructions is similar, the resulting protocols differ in important ways based on properties of the original SQ algorithm. As we show, the way that the algorithm uses randomness and the way the SQ oracle is allowed to adapt to the algorithm's internal state are key factors in building universally-sound pvCSVs.

[2]The honest prover using $\mathcal{D}_P$ may be viewed as a cheating prover with respect to the verifier holding $\mathcal{D}_V$.

pvCSVs for Deterministic SQ Algorithms. In Section 3.1, we consider SQ algorithms that do not use internal randomness. These "deterministic" SQ algorithms are considerably simpler to reason about, and as such, we obtain strong, statistically-sound pvCSVs for all such algorithms.[3]

Theorem 2. *Let $\mathcal{A}$ be a deterministic SQ algorithm that learns a concept $\mathcal{G}$. Suppose $\mathcal{A}$ makes k adaptive queries to a τ-accurate SQ oracle $\mathcal{O}$. There exists a pvCSV scheme for certifying $\mathcal{G}$ (with constant failure probability) achievable in the following complexities.*

- *Honest prover sample complexity:* $m_P \leq \tilde{O}(\sqrt{k}/\tau^2)$
- *Verifier sample complexity:* $m_V \leq O(\log(k)/\tau^2)$
- *Certificate size:* $|\pi| \leq O(k \cdot \log(1/\tau))$

In other words, even though the best learning algorithms use $\tilde{O}(\sqrt{k})$ samples to support k adaptively-chosen queries, there is a short certificate (namely, the results of the queries) that convinces a verifier with considerably less information about the distribution in $O(\log(k))$ samples. The bounds of Theorem 2 follow from the prover and verifier collectively executing a direct simulation of the original SQ algorithm with no overhead for the honest prover; in this sense, the delegation scheme can be thought of as *doubly-efficient* in its statistical complexity, à la [GKR08]. Computationally, both the prover and the verifier need to simulate the execution of the underlying SQ algorithm, so the time-complexity scales roughly linearly in the running time of the original algorithm.

pvCSVs for Randomized SQ Algorithms and SQ Protocols. In Section 4, we investigate how to build pvCSVs for SQ algorithms that use randomness. Randomness introduces significant technicality in the resulting proof systems. Here, we give an informal description of our results, including various definitions we develop to obtain the results.

Towards a pvCSV for randomized SQ algorithms, we need to reason about how the algorithm—and the (possibly-adversarial) SQ oracle—may act adaptively to the randomness. First, we say that a randomized SQ algorithm operates in ℓ "epochs" if it can be broken down into a series of ℓ deterministic SQ algorithms, which each take a fresh random string as an input. Next, we say that an SQ oracle is "public-state" if it may choose its query responses as a function of the SQ algorithm's internal randomness (in contrast to an "oblivious" oracle, which has no knowledge of the algorithm's randomness). We construct pvCSVs in the Random Oracle Model (ROM) [BR93] for all constant-epoch randomized SQ algorithms that are correct using a public-state SQ oracle.

Theorem 3. *Let $\mathcal{A}$ be a randomized SQ algorithm that learns a goal $\mathcal{G}$ with high probability. Suppose $\mathcal{A}$ makes k adaptive queries to a τ-accurate public-state SQ oracle $\mathcal{O}$, over an execution of $O(1)$ epochs. There exists a computationally-sound pvCSV scheme in the ROM for certifying $\mathcal{G}$ (with constant failure probability) achievable in the following complexities.*

- *Honest prover sample complexity:* $m_P \leq \tilde{O}(\sqrt{k}/\tau^2)$
- *Verifier sample complexity:* $m_V \leq O(\log(k)/\tau^2)$
- *Certificate size:* $\pi \leq O(k \cdot \log(1/\tau))$

In fact, Theorem 3 follows from a much more general result. We start by adapting the definition of delegation of learning [GRSY21] to the statistical query setting, yielding a format for interactive

[3] We note that such "deterministic" algorithms may still have non-deterministic behavior, due to the algorithm's adaptivity to the responses from the SQ oracle, which is assumed to be τ-accurate, but otherwise, adversarial.

learning which we call *SQ protocols*. We show that a large class of interactive SQ protocols (of which randomized SQ algorithms are a special case) can be compiled into a *canonical SQ protocol* with desirable properties. The class consists of SQ protocols where the verifier may send random and non-random challenges to the prover and may make private statistical queries (not sent to the prover) to its public-state oracle; we call these protocols "mixed-message, private-query" SQ protocols. After compilation, we obtain a canonical SQ protocol with the same completeness and soundness guarantees as the original, but which is public-coin (verifier only sends random challenges) and public-query (verifier reveals all SQs to the prover). Importantly, the canonical verifier still only makes a single batch of statistical queries.

Lemma 4. *Suppose (P, V) is a mixed-message, private-query SQ protocol where the verifier makes k adaptive queries to a τ-accurate public-state SQ oracle and verifies concept $\mathcal{G}$. There exists a canonical public-coin, public-query SQ protocol $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ that verifies $\mathcal{G}$ with the same completeness/soundness as (P, V), with the following properties.*

- $\mathsf{P}_{\mathsf{can}}$ *makes at most k additional statistical queries compared to P;*
- $\mathsf{V}_{\mathsf{can}}$ *makes a single, non-adaptive batch of k statistical queries.*

This canonical SQ protocol can then be compiled into a pvCSV via a Fiat-Shamir transformation [FS86], whose blow-up in soundness scales exponentially with the "epoch complexity" of the original protocol. The sample complexities claimed in Theorem 3 follow by giving sample-based implementations of an adaptive SQ oracle (for the prover) and non-adaptive SQ oracle (for the verifier). As a consequence, we obtain computationally-sound pvCSVs for a much broader class of learning goals—those that can be solved by an interactive SQ protocol according to Lemma 4.

As may be evident, to appropriately reason about delegation of learning in this context, we need to reason about a number of novel concepts (e.g., SQ protocols, public-coin, public-query, epoch complexity, etc.). An additional key contribution of our work is laying out precise definitions for these notions that arise in the study of proof systems for learning, presented formally in Section 4.1. We discuss our construction of pvCSVs for randomized algoirthms/SQ protocols in greater detail within the Technical Overview.

Beyond pvCSVs: The Limits of SQ Protocols. Building pvCSVs, particularly for randomized SQ algorithms, required investigating aspects of more general interactive protocols for statistical query learning. We complement our constructions of pvCSVs with a few results about the strengths and limitations of SQ protocols.

We show that, quite generically, verification of SQ protocols can be made statistically non-adaptive, in the sense that the verifier issues a single batch of statistical queries, and thus, has sample complexity scaling logarithmically in the number of queries. The exact class of protocols which we can delegate soundly in this manner is a bit technical, but it consists of a large class of "public-query" SQ protocols. In particular, the class of protocols includes private-coin protocols, where the verifier maintains secret randomness from the prover, but allows the prover to know which statistical queries the verifier issues.

Proposition 5 (Informal)**.** *For every public-query SQ protocol where the verifier makes k adaptive queries to a τ-accurate public-state SQ oracle, there is an equivalent SQ protocol where the verifier V makes a single, non-adaptive batch of queries; that is, V has sample complexity $m_V \leq O(\log(k)/\tau^2)$.*

This result shows that statistical validation of a large class of SQ protocols can be done in non-adaptive sample complexity. A natural question, then, is whether this statistically-efficient verification scheme can be made computationally-efficient. Unfortunately, we show a barrier to generic computational savings in the SQ model. Piggybacking off of a sample complexity lower bound given by [MS23] for PAC Verfication, we obtain a lower bound on the verifier's *query complexity* for the same class of SQ protocols from Proposition 5.

Corollary 6 (Informal)**.** *For every hypothesis class $\mathcal{H}$ of VC dimension d, in any SQ protocol (as in Proposition 5) that ε-PAC Verifies $\mathcal{H}$ using a τ-accurate SQ oracle for $\tau \approx \varepsilon$, the verifier makes $2^{\Omega(\sqrt{d})}$ statistical queries.*

SQ Verification under Differential Privacy. Finally, we show that the verifiers of all of our pvCSVs and interactive SQ protocols can be implemented under Differential Privacy [DMNS06]. Differential Privacy (DP) is the de facto notion to protect individuals' data in statistical analyses, but it can be statistically costly: implementing SQ algorithms under DP requires sample complexity akin to adaptive data analysis. This result shows that a verifier can validate the results of a statistical analysis much more efficiently, while still maintaining privacy over their own data set.

The new private verifiers use essentially the same number of samples as the non-private verifiers. Concretely, we state the result for the pvCSV verifiers.

Proposition 7 (Informal)**.** *Consider the pvCSV verifier from either Theorem 2 or Theorem 3. There exists an ε-DP implementation of the verifier over the samples from $\mathcal{D}_V$ (with constant failure probability) whose sample complexity scales as $O(\log(k)/\tau^2 + \log(k)/\tau\varepsilon)$.*

1.2 Technical Overview and Discussion of Results

In the remainder of the introduction, we give a more detailed overview of our models and results. Throughout, we aim to provide pointers into the main text for formal presentation. We include discussion of the significance of the results, connections to prior works, as well as possible extensions.

The primary goal of our work is to take an arbitrarily-adaptive SQ algorithm $\mathcal{A}$ and turn it into a pvCSV that can be verified efficiently. At a high-level our approach is simple: require the prover to run a direct simulation of $\mathcal{A}$; then, check the prover's work. In particular, the statistically-expensive aspect of adaptive data analysis is generating the sequence of queries $q_1, \ldots, q_k$. Once the sequence has been generated, however, the answers to the queries $q_1, \ldots, q_k$ can be checked in a single, non-adaptive batch of k statistical queries.

This observation immediately suggests a non-interactive proof system for delegating "deterministic" SQ algorithms that do not use any internal randomness, which is the focus of Section 3.1.

- The honest prover P executes $\mathcal{A}$ using their own SQ oracle $\mathcal{O}_P$ to answer any necessary queries. Along the way, the prover records the queries and results, and at the end, when $\mathcal{A}$ outputs some hypothesis h, P sends h and $\pi = \langle q_1, \mathcal{O}_P(q_1), \ldots, q_k, \mathcal{O}_P(q_k)\rangle$ to the verifier V.
- To certify that h is valid, the verifier must re-run $\mathcal{A}$, but rather than using its own oracle, it will answer any statistical queries using the answers from π. If at any point, the verifier's execution $\mathcal{A}$ requires a query q that is not provided in π, then V rejects immediately, since the

transcript sent by the prover was not consistent with the execution of $\mathcal{A}$. If the transcript is consistent, then the verifier concludes the protocol by issuing the batch of statistical queries from π to its own oracle $\mathcal{O}_V$, and checks that $\mathcal{O}_V(q_i)$ is sufficiently close to the reported $\mathcal{O}_P(q_i)$ for each query.

Note that even though we think of $\mathcal{A}$ as deterministic since it does not use randomness, its adaptivity to the SQ oracle responses introduces non-determinism, so verifying the consistency of the transcript is a non-trivial aspect of the verifier's check.

Given the simplicity of this delegation scheme, both the honest prover and verifier can be implemented very efficiently. The only overhead of the honest prover is to record the results of their statistical queries to be sent to the verifier (or posted for public verification), so the sample and time complexities scale precisely with the original complexities of the SQ algorithm $\mathcal{A}$. To answer a sequence of adaptive queries to τ-accuracy, the prover can be implemented in $\tilde{O}(\sqrt{k}/\tau^2)$ samples [BNS+16, DK22, Bla25]. The verifier also has to execute the algorithm $\mathcal{A}$, so there is no computational savings, but the statistical savings are exponential. By concentration bounds, checking a batch of k statistical queries to τ-accuracy can be achieved from $O(\log(k)/\tau^2)$ samples.

Understanding Universal Soundness. One of the key selling points of pvCSVs is the universal soundness condition: that any verifier holding $\mathcal{D}_V$ which may differ significantly from the prover's distribution $\mathcal{D}_P$ can check the certificate while maintaining soundness. As described above, our pvCSV construction does not explicitly distinguish between the prover's distribution $\mathcal{D}_P$ and the verifier's $\mathcal{D}_V$. But in a sense that can be made formal, to obtain universal soundness, we only need for standard soundness to hold from the verifier's perspective. Specifically, even if the pvCSV (h, π) was generated honestly by a prover P holding $\mathcal{D}_P$, to the verifer holding $\mathcal{D}_V$, we can imagine P to be a potential cheating prover. In this case, the verifier may reject outright, or may accept if h actually satisfies the learning goal over $\mathcal{D}_V$. The verifier's final statistical validation—paired with the correctness guarantee of SQ algorithms—ensures that if the reported query responses from π are sufficiently close to the expectations on the verifier's distribution $\mathcal{D}_V$, then h is the result of some valid invocation of the SQ algorithm $\mathcal{A}$ over $\mathcal{D}_V$.

Handling Randomness. For algorithms that use randomness, we cannot simply trust the prover to report a direct simulation of the execution of $\mathcal{A}$, using untrusted randomness. Instead, a natural idea for a pvCSV is to convert the randomized algorithm into a public-coin protocol, then apply a Fiat-Shamir transformation [FS86], to obtain a non-interactive proof. In Section 4, we show that this approach works to yield pvCSVs, but we need to be careful in how we reason about the algorithm's use of randomness, as well as how the SQ oracle affects the proof of soundness.

Starting from a randomized SQ algorithm $\mathcal{A}$, we imagine breaking the algorithm into a series of "epochs" where in the i-th epoch, the algorithm samples fresh randomness r_i, and then executes a deterministic SQ algorithm $\mathcal{A}_i$ until the end of the epoch. Every randomized algorithm can be broken into epochs, but different algorithms require more or fewer epochs. At the low extreme, an algorithm that uses a random initialization, then executes deterministically, would have epoch complexity 1; at the other extreme, a stochastic optimization algorithm that makes random choices at every iteration based on fresh independent coins will have high epoch complexity. The epoch complexity of the algorithm $\mathcal{A}$ controls the complexity of the resulting proof system.

In particular, we can turn any SQ algorithm of epoch complexity ℓ into a public-coin interactive proof in the SQ model of round complexity ℓ. At the start of each epoch i, the verifier V sends randomness r_i to the prover P. Then, the honest prover P simulates the deterministic algorithm $\mathcal{A}_i(r_i)$ using its own oracle $\mathcal{O}_P$ to answer any statistical queries. At the end of the epoch, the prover can return a transcript, similar to in the deterministic case, that summarizes the statistical queries and oracle responses. At the end of all epochs, the verifier can issue a non-adaptive batch of queries to $\mathcal{O}_V$ to ensure that the responses were all sufficiently accurate.

As described, the SQ protocol is complete, but has a subtle issue with soundness without further assumptions. In the original algorithm, $\mathcal{A}$ issues its queries to an SQ oracle $\mathcal{O}$, whereas in the protocol, the verifier V simulates $\mathcal{A}$ but delegates the SQs to the prover. The key distinction between these two setups is that the prover also receives the internal randomness of the algorithm r_i, as the public-coin message at the start of each epoch. In other words, to ensure soundness, the original randomized algorithm $\mathcal{A}$ must be correct even for SQ oracles that have full knowledge of the state of $\mathcal{A}$. Algorithms whose correctness hinges on the obliviousness of $\mathcal{O}$ to the randomness of $\mathcal{A}$ cannot be delegated in this way while maintaining soundness. Formally, we define "public-state" SQ oracles in Section 4.1, along with the corresponding correctness notion for SQ algorithms.

Once we restrict our delegation to randomized SQ algorithms which are correct under this more powerful SQ oracle, then we obtain a sound public-coin protocol. To obtain pvCSVs, we appeal to the Random Oracle Model (ROM), and show how to apply the Fiat-Shamir transform to our protocol. There is some subtlety in defining the notion of universal soundness for pvCSVs in the ROM and then subsequently arguing that Fiat-Shamir applied to our protocol obtains such soundness. With the appropriate definitions in place, the soundness proof follows by following the state restoration approach of [BSCS16, CY24].

Beyond SQ Algorithms. Building pvCSVs for randomized SQ algorithms required us to define and investigate a number of more general models of proof systems in the SQ model. A key result we show in Section 4.2 is that a diverse class of interactive SQ protocols can be compiled into a canonical public-coin SQ protocol. In a sense, this result is an analogue of the established understanding of public-coin protocols for delegation of computation. We show that, quite generically, if an SQ protocol consists of a verifier that reveals its randomness to the prover, there is a canonical version of the protocol where the only messages the verifier sends to the prover are its randomness. Further, the verifier need not maintain any private statistical queries, but can delegate all of its queries to the prover, then execute one final non-adaptive validation. This result holds in the same model of public-state SQ oracle, which may adapt to the state of the verifier. With this canonical compiler for SQ protocols, we can similarly obtain computationally-sound pvCSVs in the ROM, by applying Fiat-Shamir to the canonical protocol.

Towards Computationally-Efficient Certification of Learning. In this work, our focus is on developing statistically-efficient proofs of learning. We make no effort to optimize the verifier's running time. Our pvCSV verifier for an SQ algorithm $\mathcal{A}$ runs in time proportional to the running time of $\mathcal{A}$. A natural question in the study of proof systems is whether we can save on computation time during verification.

In Section 5, we show some barrier to generic speed-ups (at least within the SQ framework). In

particular, Corollary 6 shows that there are statistically learnable VC classes, for which there is no computationally-efficient SQ verification scheme, even if we allow for interaction. Our SQ lower bound is actually a consequence of the *efficiency* of our verification schemes (in terms of the number of statistical queries) and an existing lower bound on the samples required for verification from [MS23] in terms of the VC dimension. In other words, a fast SQ verification scheme (which doesn't make many SQ queries) would imply an impossibly-statistically-efficient verification scheme for the VC class studied by [MS23].

Nevertheless, in some sense, our exploration of pvCSVs and SQ protocols provides new design principles for simultaneously statistically and computationally efficient verification. Recall that the canonical protocol of Lemma 4, which compiles into a pvCSV as in Theorem 3, applies not just to randomized algorithms but to a broad class of SQ protocols. Further, the verifier in the resulting pvCSV runs in time proportional to (or less than) the verifier in the original protocol—not the full simulation of (P, V). In other words, if we can design SQ protocols that reduce verification time, then we may get simultaneous statistical and computational efficiency for free (provided the protocol is covered by Lemma 4).

Concretely, our results show that minimizing the epoch complexity of a learning procedure—even if that entails designing an interactive proof system—may actually lead to more efficient pvCSVs than simply focusing on delegating algorithms for learning. Such observations may also motivate a deeper study of pseudorandomness in machine learning; for instance, if a randomized SQ algorithm can actually be proved correct under a weaker source of randomness, it may reduce the epoch complexity and improve efficiency.

Publicly-Verifiable End-to-End DP. Finally, in Section 6, we show Proposition 7, which implies a mechanism for verifying pvCSVs statistically efficiently under Differential Privacy with respect to the verifier's samples. Given such a verifier, it's tempting to wonder whether we can achieve End-to-End DP over both the prover's samples from $\mathcal{D}_P$ and the verifier's from $\mathcal{D}_V$. The existence of a DP verifier implies an honest prover strategy to achieve such a goal. Due to the connections between adaptive data analysis and DP, this honest prover strategy is no more expensive than accounting for adaptivity.

That said, in our setting, we'd have no guarantees about the behavior of the cheating prover, who might violate DP arbitrarily. To handle the possibility of a cheating prover, we could employ a scheme for Certified DP of [BGKW24]. Certified DP ensures that a prover's release of statistical queries follows a DP mechanism with respect to some committed-to, but untrusted database $X \sim \mathcal{D}_P$, in a manner that can be verified publicly. In this sense, adding our DP statistical validation scheme as post-processing to the output of a Certified DP mechanism would allow the verifier to be convinced of End-to-End DP, while also ensuring statistical accuracy of queries from the untrusted database X under the verifier's distribution $\mathcal{D}_V$.

1.3 Related Works

Our study of pvCSVs and SQ protocols lives in the intersection of learning theory and cryptographic proof systems. We highlight some of the most relevant related lines of work.

Proof Systems for Learning and Statistics. Proof systems for statistical algorithms grew out of the literature on interactive (cryptographic) proof systems [GMR85, Bab85]. The PAC Verification, introduced in [GRSY21], led to a number of results about strengths and limitations of interactive proofs of learning [MS23, GJK+24]. Most related to our work is the work of Mutreja and Shafer [MS23], who also thought about the issue of delegating SQ algorithms in the context of loss minimization. In fact, in a subsequent journal version [MS26], the authors independently include a result analogous to Proposition 5, on the interactive delegation of adaptive SQ algorithms.

Other works have investigated certifying properties of learning algorithms. [BGKW24] recently designed mechanisms for public cerification of differential privacy (and other private probabilistic mechanisms) in the release of statistical queries. Following this work, [BTFM+25] extended the techniques to give a publicly-verifiable implementation of DP Stochastic Gradient Descent, without appealing to heavy-handed cryptographic primitives for certifying generic computations.

Beyond proofs for learning algorithms, there has been significant work investigating proof systems for properties of distributions. Initiated by [CG18], a sequence of works [HR22, HR23, HR24b, HR24a] has established doubly-efficient proof systems for distribution testing.

Adaptive Data Analysis. Since its identification as a key algorithmic challenge [DFH+15b, DFH+15a, DFH+15c], Adaptive Data Analysis has seen significant developments, in line with the developments of sophisticated tools for Differential Privacy [DMNS06]. The majority of work studying adaptive data analysis focuses on the problem of answering statistical queries. [BNS+16] first showed near-optimal $\tilde{O}(\sqrt{k})$ dependence on the number of adaptively chosen queries. Subsequently, a sequence of papers has simplified their analysis and techniques and improved their bounds [FS18, JLN+19, DK22]. Recently, [Bla25] showed that a much simpler subsampling mechanism actually suffices for optimal adaptive data analysis—despite the fact that it does not suffice for DP. Indeed, the analysis of [Bla25] does not go through the standard "transfer theorem" that DP implies adaptive generalization, but rather analyzes the effects of subsampling directly.

2 Preliminaries

We give formal definitions of the statistical query model of learning [Kea98] that we adopt, as well as proof systems developed in the context of verifying machine learning [GRSY21, MS23]. We start with background on the SQ Model. Then, we discuss our formalisms for algorithms and proof systems that use SQ oracles. We defer some technical aspects of our learning model (particularly those related to algorithms' use of randomness) to the relevant technical section (Section 4.1). Finally, we review background on adaptive data analysis and the complexity of implementing SQ algorithms from samples [DFH+15a].

2.1 The Statistical Query Model

The Statistical Query (SQ) Model [Kea98] abstracts away the notion of learning from samples to the notion of learning from approximate statistics. In this model, the learning algorithm is allowed

to specify predicates $q : \mathcal{X} \to \{0,1\}$ from a collection[4] $\mathcal{Q} \subseteq \{0,1\}^{\mathcal{X}}$ and receive—from the *statistical query oracle*—the (rough) expectation of the queried predicate on the distribution of interest $\mathcal{D}$.

Definition 2.1 (SQ Oracle, idealized)**.** *A Statistical Query (SQ) Oracle is a stateful algorithm $\mathcal{O} : \mathcal{Q} \to [0,1]$ that takes input queries $q \in \mathcal{Q}$ and responds with evaluations $\mathcal{O}(q) \in [0,1]$. For $\tau > 0$, the oracle $\mathcal{O}$ is τ-accurate over $\mathcal{D}$ if for any finite sequence of $k \in \mathbb{N}$ (adaptively-selected) queries $q_1, q_2, \ldots, q_k$, for all $i \in [k]$*

$$\left| \mathcal{O}(q_i) - \mathop{\mathbf{E}}_{X \sim \mathcal{D}}[q_i(X)] \right| \leq \tau.$$

Many notable learning algorithms can be described in the SQ Model, including most (Agnostic) PAC Learning algorithms[5] [Kea98] as well as more modern ML algorithms like Gradient Descent.

We remark that, per the definition, the SQ oracle $\mathcal{O}$ with τ-accuracy must always report approximate expectations within τ of the true expectation. As such, given access to an SQ oracle, we can hope to design algorithms that always succeed. In Section 2.3, we discuss concrete, sample-based implementations of the SQ oracle abstraction, which necessarily introduce a failure probability.

Learning Goal. We consider an abstract setting of learning over distributions $\mathcal{D}$ supported on a domain $\mathcal{X}$, such as $\mathcal{X} = \{0,1\}^d$ for some finite dimension $d \in \mathbb{N}$. Additionally, we consider an abstract collection of hypotheses $\mathcal{H}$. We define the goal of learning in terms of identifying a hypothesis within some "good" set of hypotheses $\mathcal{G}(\mathcal{D}) \subseteq \mathcal{H}$, parameterized by the distribution $\mathcal{D}$.

Definition 2.2 (Learning goal, abstract)**.** *Fix a class of hypotheses $\mathcal{H}$ and a learning goal $\mathcal{G} \subseteq \mathcal{H}$. For a distribution $\mathcal{D}$, we say an algorithm $\mathcal{A}$ learns $\mathcal{G}$ if $\mathcal{A}$ outputs some $g \in \mathcal{G}(\mathcal{D})$ in the target set of hypotheses.*

For instance, in the context of Agnostic PAC learning, we can take $\mathcal{H}$ to be the concept class $\mathcal{H} \subseteq \{h : \{0,1\}^d \to \{0,1\}\}$, and the good subset $\mathcal{G}(\mathcal{D}) \subseteq \mathcal{H}$ to be all hypotheses achieving classification error $\mathbf{Pr}[g(x) \neq y] \leq \mathbf{Pr}[h^*(x) \neq y] + \varepsilon$ competitive with the best $h^* \in \mathcal{H}$ over $\mathcal{D}$. We adopt this abstract notion of learning, rather than a concrete notion like the PAC framework, in order to emphasize the generality of our approach. At the extreme, we may take the hypotheses $\mathcal{H}$ to be the collection of sequences of statistical queries and expectations, and the good set $\mathcal{G}(\mathcal{D})$ to be sequences that arise as a valid execution of a statistical algorithm on $\mathcal{D}$. (We comment further on our choice of abstract Learning Goal after Definition 2.7 of protocols for delegating SQ learning.)

Statistical Query Algorithms. Naturally, we define SQ algorithms as algorithms that may make calls to an SQ oracle. For SQ algorithms that solve a learning goal, we need to specify the approximation parameter $\tau > 0$ necessary to guarantee correctness. Formally, we distinguish between deterministic and randomized SQ algorithms.

[4] In our work, we take $\mathcal{Q} = \{0,1\}^{\mathcal{X}}$ to be the set of all boolean functions, so often drop explicit reference to $\mathcal{Q}$. In this work, we will not consider the computational complexity of evaluating functions $q \in Q$, and instead, measure the complexity of learning in terms of the total number/sequence of queries issued.

[5] Learning Parities is the most notable example of a task that is PAC learnable, but not SQ learnable.

Definition 2.3 (SQ Learning, deterministic)**.** *Fix a learning goal $\mathcal{G}$ and $\tau > 0$. A deterministic algorithm $\mathcal{A}$ τ-SQ learns $\mathcal{G}$ if, for any distribution $\mathcal{D}$ and τ-accurate SQ oracle $\mathcal{O}$ for $\mathcal{D}$, $\mathcal{A}^{\mathcal{O}}$ outputs a hypothesis $g \in \mathcal{G}(\mathcal{D})$.*

When $\mathcal{A}$ is deterministic, without loss of generality, we can think of the SQ oracle $\mathcal{O}$ as a pre-specified function where $\mathcal{O}(q)$ is defined for all $q \in \mathcal{Q}$ up front, rather than a stateful algorithm responding sequentially. Specifically, if the queries that $\mathcal{A}$ issues are a deterministic function of the input to $\mathcal{A}$ and the responses given by $\mathcal{O}$ so far, then a pre-specified function can simulate any stateful oracle, by simply running $\mathcal{A}$ ahead of time. Note the output of running $\mathcal{A}$ with τ-accurate oracle $\mathcal{O}$ may still be nondeterministic based on the choice of $\mathcal{O}$, even if $\mathcal{A}$ doesn't flip coins. That said, we insist that given a τ-accurate (idealized) oracle $\mathcal{O}$, the algorithm $\mathcal{A}$ always succeeds.

Randomized SQ Algorithms. In our study of SQ proof systems, the distinction between deterministic and randomized algorithms is significant. In contrast to deterministic algorithms, when $\mathcal{A}$ is randomized, it becomes important to think of the SQ oracle $\mathcal{O}$ as a stateful adversary, who responds with knowledge of prior queries/responses, subject to τ-accuracy. That is, for a sequence of queries $q_1, q_2, \ldots, q_k$, the response $a_i = \mathcal{O}(q_i)$ may depend on $\langle q_1, a_1, q_2, a_2, \ldots, q_{i-1}, a_{i-1}, q_i \rangle$. An SQ algorithm that learns $\mathcal{G}$ must output a good hypothesis with high probability, no matter what decisions the oracle makes in response to the sequence of queries.

Definition 2.4 (SQ Learning, randomized)**.** *Fix a learning goal $\mathcal{G}$ and $\tau > 0$, $\gamma > 0$. A randomized algorithm $\mathcal{A}$ (τ, γ)-SQ learns $\mathcal{G}$ if for any distribution $\mathcal{D}$, and any τ-accurate SQ oracle $\mathcal{O}$ for $\mathcal{D}$, $\mathcal{A}^{\mathcal{O}}$ outputs a hypothesis $g \in \mathcal{G}(\mathcal{D})$ with probability at least $1 - \gamma$ over the random coins of $\mathcal{A}$.*

With this high-level definition in place, we defer significant details and definitions about randomized SQ algorithms to Section 4.1, which are essential to understanding our pvCSVs for randomized SQ algorithms and SQ protocols. In particular, we consider two flavors of SQ oracles for randomized algorithms, "oblivious" oracles (as in Definition 2.1) and "public-state" oracles (see Definition 4.3).

Complexity Measures for SQ algorithms. An important complexity measure of SQ algorithms is the number of statistical queries the algorithm makes B, as well as the *adaptivity* to prior query responses k. The adaptivity of an SQ algorithm $\mathcal{A}$ is the number of rounds in which batches of queries are issued to $\mathcal{O}$. At the extremes of adaptivity, $k = B$ is a fully-adaptive algorithm, where the choice of each query q_i is determined as an arbitrary function of the prior queries and responses $\langle q_1, \mathcal{O}(q_1), q_2, \mathcal{O}(q_2), \ldots, q_{i-1}, \mathcal{O}(q_{i-1}) \rangle$; whereas $k = 1$ is a *non-adaptive* SQ algorithm and all of the queries are issued in a single batch. As we discuss in Section 2.3, the adaptive query complexity determines how efficiently we can implement the SQ oracle from samples.

Definition 2.5 (Query Complexity)**.** *Fix $\tau > 0$. An SQ algorithm has τ-query complexity (k, B) if for all distributions $\mathcal{D}$ and τ-accurate SQ oracles $\mathcal{O}$, the execution of $\mathcal{A}^{\mathcal{O}}$ issues at most B queries over k rounds of adaptivity.*

Additionally, we will informally track the running time of SQ algorithms (eliding details of computation over real-valued responses from the SQ oracle).

Definition 2.6 (Running Time)**.** *Fix $\tau > 0$. An SQ algorithm has running time $T(\tau)$ if over all $\mathcal{D}$ and all τ-accurate SQ oracles $\mathcal{O}$, the execution of $\mathcal{A}^{\mathcal{O}}$ runs in at most $T(\tau)$ steps.*

For simplicity's sake, in both query and time complexity, we assume that randomized SQ algoirthms provide a deterministic guarantee on the complexity (that is, a BPP-style guarantee).

2.2 Proof Systems for Statistical Learning

We consider proof systems for statistical learning based off of the classic notions from complexity theory and cryptography [GMR85, Bab85], and more recently the notion of PAC Verification [GRSY21]. In this setting, a prover P and a verifier V interact in order for the verifier to be convinced that a given hypothesis $g \in \mathcal{G}(\mathcal{D})$ is good for a learning goal. We model P and V as randomized algorithms and denote their interaction as (P, V). We adapt the notions from recent works on PAC Verification to define a model of interactive proof system where the parties have access to SQ oracles as follows.

Definition 2.7 (SQ Protocol, adapted from [GRSY21, MS23])**.** *A Statistical Query protocol is given by the interaction of two randomized SQ algorithms, the honest prover P and the verifier V, denoted (P, V).*

Fix a learning goal $\mathcal{G}$, $\tau_P, \tau_V > 0$, and $\gamma > 0$. An SQ protocol (P, V) (τ_P, τ_V, γ)-SQ-verifies $\mathcal{G}$ if for any distribution $\mathcal{D}$ and τ_P-accurate SQ oracle $\mathcal{O}_P$ and τ_V-accurate SQ oracle $\mathcal{O}_V$, the following conditions hold:

- *Completeness: the honest protocol $(P^{\mathcal{O}_P}, V^{\mathcal{O}_V})$ outputs $g \in \mathcal{G}(\mathcal{D})$ with probability at least $1-\gamma$ over the random coins of P and V.*
- *Soundness: for any cheating prover strategy $\tilde{P}$, with probability at least $1-\gamma$ over the random coins of V, the protocol $(\tilde{P}^{\mathcal{O}_V}, V^{\mathcal{O}_V})$ outputs $g \in \mathcal{G}(\mathcal{D})$ or V rejects.*

For failure probability $\delta > 0$, the SQ protocol has sample complexity (m_P, m_V) if the honest prover's oracle $\mathcal{O}_P$ can be implemented from at most m_P samples and the verifier's oracle $\mathcal{O}_V$ can be implemented from at most m_V samples, with probability at least $1 - \delta$ over i.i.d. samples from $\mathcal{D}$.

In other words, in an SQ protocol, there exists an honest prover strategy that allows the verifier to accept a good hypothesis $g \in \mathcal{G}(\mathcal{D})$ with high probability, and conversely, if the verifier accepts a hypothesis $\tilde{g}$, then with high probability,[6] the hypothesis $\tilde{g} \in \mathcal{G}(\mathcal{D})$, even if it was generated through an interaction with a different prover.

Note that, as is standard, we imagine the cheating prover $\tilde{P}$ is all-powerful and may have arbitrary knowledge of the distribution $\mathcal{D}$. Additionally, we equip the cheating prover *with the verifier's oracle* $\mathcal{O}_V$; that is, for any query that the verifier asks $q \in \mathcal{Q}$, the precise value $a = \mathcal{O}_V(q)$ is known to both V and $\tilde{P}$. (That said, the verifier may choose to keep queries private from the prover.)

[6] For simplicity, we use a single parameter γ for the completeness and soundness error. Of course, it may also be interesting to consider protocols that achieve different completeness and soundness parameters. In fact, all of our protocols achieve completeness 1 in the SQ oracle model.

There are many properties of SQ protocols that may be of interest. For instance, we consider both private- and public-coin protocols (or, as hinted above, *private-* and *public-query* protocols). We defer these definitions—particularly those related to our construction of pvCSVs—to Section 4.1.

Implementing the oracles from samples will necessarily incur some additional failure probability $\delta > 0$, distinguished from soundness error γ of the original protocol. Accounting for the failure probability γ of the oracle-based protocol separately from the failure probability δ that arises from sampling actually leads to improved analysis of the soundness of some of our protocols. We discuss background on the sample complexity of adaptive SQ algorithms in Section 2.3.

Comparison to PAC Verification. Definition 2.7 is directly inspired by the notion of PAC Verification of statistical algorithms [GRSY21, MS23], but departs in a few key ways.

First off, our notion of learning is not tied directly to Agnostic PAC learning (or loss minimization as in [MS23]). Instead, we elect to use our abstract learning goal (Definition 2.2) as the basis for SQ Protocols. We make this choice because the protocols we design are not actually tied to any properties of loss minimization, but instead run a direct simulation of a given SQ algorithm. So, provided we start with a learning algorithm $\mathcal{A}$ that achieves its learning goal with good probability, we can turn it into a SQ protocol that also learns $\mathcal{G}$.[7]

Second, we define SQ protocols in the SQ oracle model, rather than in terms of direct samples from the distribution. In reality, we will be interested in understanding the statistical resources (i.e., samples) necessary to implement the verifier and honest prover strategies. As in the earlier works on PAC verification, the goal is to design SQ protocols where the verifier's oracle $\mathcal{O}_V$ can be implemented more efficiently than the honest prover's $\mathcal{O}_P$. For instance, quantitatively, perhaps P requies an oracle with much tighter tolerance than V, $\tau_P < \tau_V$; or as in our work, qualitatively, P may require an adaptive oracle, whereas V uses a non-adaptive oracle. That said, nothing precludes a hybrid definition, where the verifier only has SQ oracle-access to $\mathcal{D}$, but the prover has more refined access through samples or point evaluations..

Computational Soundness and The Random Oracle Model. Definition 2.7 assumes that cheating provers are computationally-unbounded. We refer to this property as *statistical soundness*. In contrast, we say that *computational soundness* holds when we prove soundness against probabilistic polynomial-time cheating provers $\tilde{P}$. When we consider computational soundness, we will still assume that $\tilde{P}$ may hold prior, detailed knowledge of $\mathcal{D}$ (so need not be given an SQ oracle), but does not have the computational resources to break cryptographic primitives.

We will be particularly interested in computational soundness when designing pvCSVs for randomized SQ algorithms. To do so, we leverage a key cryptographic paradigm: The Fiat-Shamir Transformation [FS86]. The heuristic takes a public-coin interactive protocol and removes interaction by emulating the verifier's random messages with calls to a cryptographic hash function. Classically, the soundness of Fiat-Shamir is proved in the Random Oracle Model (ROM) [BR93], which models public access to an (idealized) hash function.

[7] One may even be inclined to define delegation of learning as a distributional simulation of a given algorithm $\mathcal{A}$. In fact, once formalized, many of our protocols would satisfy such a notion, but we encounter issues with soundness when we want to achieve non-interactive protocols (i.e., pvCSVs) for randomized SQ algorithms in Section 4.

Definition 2.8 (Random Oracle Model)**.** *For output size $m \in \mathbb{N}$, the Random Oracle Model augments the base computational model by assuming that all parties have access to a public, uniformly-random function $f : \{0,1\}^* \to \{0,1\}^m$ that can be evaluated at unit cost.*

Note that when we consider protocols that operate in the ROM, the probability of violating computational soundness is over the verifier's random coins as well as the draw of the random oracle.

While, in full generality, soundness does not transfer from the ROM to realizable models of computation [Bar01,GK03], proofs of soundness in the ROM are generally considered as strong evidence of security and sufficient for many practical applications. Our pvCSVs for randomized SQ algorithms will be proved computationally-sound in the ROM.

2.3 Sample-Based Implementation of SQ Oracles

To run SQ learning algorithms/protocols, we need a concrete implementation of the SQ oracle abstraction based on samples drawn from $\mathcal{D}$. The number of samples required to guarantee statistical validity of the oracle's responses depends on features of the SQ algorithm $\mathcal{A}$.

In particular, the sample complexity depends on the number of queries issued B, the desired accuracy τ, as well as the *adaptivity* k. The adaptivity of an SQ algorithm $\mathcal{A}$ is the number of rounds in which batches of queries are issued to $\mathcal{O}$. At the extremes of adaptivity, $k = B$ is a completely adaptive algorithm, where the choice of each query q_i is determined as an arbitrary function of the prior queries and responses $\langle q_1, \mathcal{O}(q_1), q_2, \mathcal{O}(q_2), \ldots, q_{i-1}, \mathcal{O}(q_{i-1})\rangle$; whereas $k = 1$ is a *non-adaptive* SQ algorithm and all of the queries are issued in a single batch. The following definition captures these dependencies.

Definition 2.9 (SQ Oracle implementation)**.** *An algorithm $\mathcal{O}$ is a (k, B, τ, δ)-implementation of an SQ oracle for a distribution $\mathcal{D}$, if $\mathcal{O}$ can support any sequence of queries $q_1, q_2, \ldots, q_B$ satisfying the following properties:*

- *Rounds of adaptivity k: the sequence of queries is issued in at most k batches; queries selected in the ith batch may depend arbitrarily on the queries and responses of the first $i-1$ batches*
- *Query budget B: the total number of queries in the sequence is upper bounded by B*
- *Accuracy τ: for every query q in the sequence, the response $\mathcal{O}(q)$ is τ-accurate on $\mathcal{D}$.*
- *Failure probability δ: for all such query sequences of complexity (k, B), with probability at least $1 - \delta$, all queries are answered τ-accurately.*

The sample complexity of an SQ oracle implementation is the number $m \in \mathbb{N}$ of i.i.d. samples from $\mathcal{D}$ required to guarantee (k, B, τ, δ)-implementation.

We define the notion of an SQ Oracle implementation in terms of the rounds of adaptivity and the total number of queries, because certain mechanisms are able to exploit limited adaptivity to achieve better sample complexity [HR10, DFH+15c]. Note that, in order to guarantee τ-accurate queries, we can assume that our implementation returns values using $O(\log(1/\tau))$ bits of precision.

Crucially for our work, the sample complexity required to answer a non-adaptive batch of queries is an exponential improvement over the complexity required to answer an adaptively-selected sequence of queries. For a non-adaptive batch of queries, the empirical SQ oracle that takes m samples $x_1, \ldots, x_m \sim \mathcal{D}$ and reports $\mathcal{O}(q) = \frac{1}{m}\sum_{i=1}^{m} q(x_i)$ achieves logarithmic dependence on the number of queries.

Proposition 2.10 (Non-adaptive Sample Complexity)**.** *For any $\tau, \delta > 0$, the empirical oracle is a $(1, B, \tau, \delta)$-implementation of an SQ oracle for $\mathcal{D}$ with sample complexity*

$$m \leq O\left(\frac{\log(B/\delta)}{\tau^2}\right).$$

Proposition 2.10 follows by a standard application of Hoeffding's inequality. The sample complexity for answering an adaptively-selected sequence of statistical queries, however, requires much more sophisticated analysis. The complexity was only established in the past decade, after connecting the problem of adaptive data analysis with the technique of Differential Privacy [DMNS06].

In the case of fully-adaptive algorithms, where $k = B$ statistical queries are issued adaptively, the (roughly) optimal sample complexity scales with $\sqrt{k}$, rather than $\log(k)$. Initially, the upper bounds followed by leveraging the stability properties of differentially-private query release [BNS+16,DK22], but the most recent result of [Bla25] leverages a subsampling technique without DP.[8]

Proposition 2.11 (Theorem 3 of [Bla25], Adaptive Sample Complexity Upper Bound)**.** *For any $\tau, \delta > 0$, there exists a mechanism that is a (k, k, τ, δ)-implementation of an SQ oracle for $\mathcal{D}$ with sample complexity*

$$m \leq O\left(\frac{\sqrt{k \cdot \log(k/\delta) \cdot \log(1/\delta)}}{\tau^2}\right)$$

The lower bound holds based on the construction of (interactive) fingerprinting codes [HU14,SU15], and holds unconditionally in large-domain settings and assuming the existence of one-way functions in all settings.

Proposition 2.12 (Theorems 1 & 2 of [SU15], Adaptive Sample Complexity Lower Bound)**.** *For any $\tau \in (0, 0.49)$, any mechanism which is a $(k, k, \tau, 1/2)$-implementation of an SQ oracle for $\mathcal{D}$ has sample complexity*

$$m \geq \Omega(\sqrt{k})$$

if either of the following assumptions hold: one-way functions exist, and the mechanism is computationally bounded; or the space of samples $\mathcal{X}$ is sufficiently large: $|\mathcal{X}| \geq 2^{O(k)}$.

In other words, there is a *provable* exponential gap in the sample complexity required to answer k queries adaptively versus non-adaptively.

[8] Note that the bounds achieved by [DK22] and [Bla25] are technically incomparable, but differ only in poly-logarithmic factors in k and τ. Either mechanism could be used to implement our honest prover's SQ oracle.

3 Publicly-Verifiable Certificates of Statistical Validity

In this section, we introduce the main conceptual contribution of this work—*publicly-verifiable Certificates of Statistical Validity*—which allows a prover to publish a digest from the execution of a learning algorithm that *any* verifier can subsequently test for statistical validity. Importantly, the verifier of a publicly-verifiable Certificate of Statistical Validity (pvCSV) need not hold the same distribution as the prover. Instead, the verifier can test for statistical validity with respect to their own distribution $\mathcal{D}_V$, through an SQ oracle or samples.

To formalize pvCSVs, we must first introduce the notion of soudness that makes proofs "publicly-verifiable." We define *universal soundness* to ensure soundness holds no matter what distribution the verifier holds.

Definition 3.1 (Universal Soundness for SQ protocols)**.** *Fix a learning goal $\mathcal{G}$, $\tau_V > 0$, and $\gamma > 0$. An SQ protocol (P, V) has γ-universal soundness if for any verifier distribution $\mathcal{D}_V$ and τ_V-accurate SQ oracle $\mathcal{O}_V$ for $\mathcal{D}_V$, and for any prover $\tilde{P}$, with probability at least $1-\gamma$, $(\tilde{P}^{\mathcal{O}_V}, V^{\mathcal{O}_V})$ outputs $g \in \mathcal{G}(\mathcal{D}_V)$ or V rejects.*

It is easy to see that universal soundness of SQ Protocols is actually just a restatement of standard soundness; syntactically, our definition simply renames the distribution $\mathcal{D}$ to be in terms of $\mathcal{D}_V$. But importantly, the restatement of the property allows us to reason rigorously about proof systems when the prover and verifier hold different distributions. Under universal soundness, when the verifier has access to $\mathcal{D}_V$, running the protocol results in V rejecting or V accepting a hypothesis that is good for the verifier's distribution, regardless of the prover's distribution $\mathcal{D}_P$.

In a bit more detail, we can imagine an honest prover with oracle access to $\mathcal{O}_P$ for a distribution $\mathcal{D}_P$, while a verifier has access to an oracle O_V for a completely different distribution $\mathcal{D}_V$. In effect, we can view the honest prover $P^{\mathcal{O}_P}$, as a possible cheating prover for any downstream verifier $V^{\mathcal{O}_V}$. Still, if the verifier accepts the proof of some non-$\perp$ hypothesis, then universal soundness guarantees that $h \in \mathcal{G}(\mathcal{D}_V)$. The appeal of universal soundness shows itself when we consider non-interactive proofs, where a prover wants to publish a fixed certificate, which any verifier (holding any $\mathcal{D}_V$) can verify for themselves. In this context, we can define pvCSVs as a non-interactive SQ protocol satisfying universal soundness.

Definition 3.2 (pvCSV)**.** *Fix a learning goal $\mathcal{G}$, $\tau_P, \tau_V > 0$, and $\gamma > 0$. A publicly-verifiable Certificate of Statistical Validity (pvCSV) scheme is a non-interactive SQ protocol, where a prover P with τ_P-accurate SQ oracle for distribution $\mathcal{D}_P$ publishes a hypothesis $h \in \mathcal{H}$ with certificate π. The pvCSV (τ_P, τ_V, γ)-certifies $\mathcal{G}$ if any verifier V with τ_V-accurate SQ oracle $\mathcal{O}_V$ for distribution $\mathcal{D}_V$ can read (h, π) and accept or reject with the following guarantees:*

- *Completeness: if $\mathcal{D}_P = \mathcal{D}_V$, the honest prover, with τ_P-accurate SQ oracle for $\mathcal{D}_P$, can generate a hypothesis-certificate pair $(h, \pi) \leftarrow P^{\mathcal{O}_P}$ such that $V^{\mathcal{O}_V}$ accepts and $h \in \mathcal{G}(\mathcal{D}_V)$ with probability at least $1-\gamma$, over the random coins of P and V.*
- *Universal Soundness: for any verifier distribution $\mathcal{D}_V$, for any prover strategy $(h, \pi) \leftarrow \tilde{P}^{\mathcal{O}_V}$, $V^{\mathcal{O}_V}$ rejects or $h \in \mathcal{G}(\mathcal{D}_V)$ with probability at least $1-\gamma$ over the random coins of V.*

The construction of a pvCSV allows a prover with access to a distribution $\mathcal{D}_P$ to execute an algorithm $\mathcal{A}$ that learns $\mathcal{G}$ once and publish the results (h, π). Then, any party who is interested in

running $\mathcal{A}$ on their own distribution $\mathcal{D}_V$ can, instead, run the verification algorithm V on (h, π). If (h, π) is accepted, then by universal soundness, $h \in \mathcal{G}(\mathcal{D}_V)$ is guaranteed to be valid for the verifier's distribution.

Remarks. A few remarks about the definition of pvCSVs are in order.

- *Distributional Access:* We describe our pvCSVs as SQ protocols, then give sample-based implementations. That said, pvCSVs could equally encompass non-interactive protocols for verification of learning, with more general distributional access, provided we define a more general notion of universal soundness. As in [GRSY21], the type of statistical access available may serve as a qualitative difference in the complexity of the prover and verifier.
- *Complexity Measures:* Quantitatively, there are many measures of complexity one could track for pvCSVs. We will be most interested in the sample complexity (using i.i.d. random draws) of the honest prover and verifier from $\mathcal{D}_P$ and $\mathcal{D}_V$, respectively. In general, we may also track other natural complexity measures, such as the time complexity or size of the certificate.
- *Computational Soundness:* Definition 3.2 considers universal soundness with respect to computationally-unbounded provers. We also consider computationally-sound pvCSVs, where soundness only holds with respect to PPT algorithms. Concretely, we will prove soundness in the Random Oracle Model, where the failure probability is taken over the random draw of random oracle (in addition to the verifier's coins).
- *On Robust Completeness?* Note that in our notion of completeness, we assume that the honest prover and verifier have access to exactly the same distribution $\mathcal{D}_P = \mathcal{D}_V$. Given the motivating perspective on pvCSVs, we might hope for a weaker notion of distributional closeness required for completeness. Such a guarantee may be formally defined in future work. We remark that—paired with universal soundness—we still get some notion of robustness. If the verifier accepts (h, π) then the distributions are indistinguishable, according to the queries asked by (a specific execution of) $\mathcal{A}$, regardless of how close they are in any standard notion of statistical/computational indistinguishability.

3.1 A pvCSV for Every Deterministic SQ Algorithm

Our first construction establishes that every deterministic SQ algorithm has a statistically-sound pvCSV where the verifier answers k queries non-adaptively and has sample complexity $O(\log(k))$, whereas the best known, efficient prover must answer k possibly adaptive queries with sample complexity scaling with $\tilde{O}(\sqrt{k})$. We present the pvCSV scheme as a non-interactive SQ protocol, Protocol 1. To generate the certificate, the honest prover first simulates the execution of $\mathcal{A}$ using the prover's SQ oracle $\mathcal{O}_P$, returning a hypothesis h. Along the way, the prover writes down the list of queries $\boldsymbol{q}$ and answers $\mathcal{O}_P(\boldsymbol{q})$ provided by the oracle as the certificate. Then, the verifier also simulates the execution of $\mathcal{A}$, but uses the digest of query-answer pairs $(\boldsymbol{q}, \mathcal{O}_P(\boldsymbol{q}))$ provided by the certificate, *rather than making any adaptive statistical queries.* Finally, the verifier runs a *batch* statistical validation of the answers provided in the certificate using its oracle $\mathcal{O}_V$ for $\mathcal{D}_V$. The key insight is that the verifier can non-adaptively check every (adaptively-chosen) query produced during the execution of $\mathcal{A}$.

Protocol 1: pvCSV for Determinstic SQ Algorithm

Setup: Let $\mathcal{G}, \tau$, γ and $\mathcal{A}$ be as described in Theorem 3.4.
Let $\mathcal{O}_P$ be a $\tau/3$-accurate SQ oracle for $\mathcal{D}_P$.
Let $\mathcal{O}_V$ be a $\tau/3$-accurate SQ oracle for $\mathcal{D}_V$.

Simulation of SQ algorithm $\mathcal{A}$
Prover P:
Initialize an empty transcript $\pi = \langle\rangle$
Simulate $\mathcal{A}$ by executing $\mathcal{A}^{\mathcal{O}_P}$; specifically:
- whenever $\mathcal{A}$ issues an SQ q, P queries $\mathcal{O}_P(q)$ and updates $\pi \leftarrow \langle \pi, q, \mathcal{O}_P(q) \rangle$

if *simulation of $\mathcal{A}$ returns h* **then**
  P sends (h, π) to V

Verification of transcript
Verifier V:
Read and typecheck (h, π). Parse π into $(\boldsymbol{q}, \boldsymbol{a})$ as follows:
- $\boldsymbol{q} := (q_1, q_2, \ldots, q_{B_V})$, the list of SQs issued during simulation of $\mathcal{A}$
- $\boldsymbol{a} := (a_1, a_2, \ldots, a_{B_V})$, the answers to $\boldsymbol{q}$, where a_j should equal $\mathcal{O}_P(q_j)$

Check that π is consistent with $\mathcal{A}$ outputting h; that is,
- Simulate $\mathcal{A}$ using $\boldsymbol{a}$ to answer SQs
- Check that $\mathcal{A}$ returns $h \neq \perp$

if *π is NOT consistent with $\mathcal{A}$* **then**
  Reject and output $\perp$

Statistical Validation of simulated oracle queries
if *NOT* $\mathsf{Validate}(\boldsymbol{q}, \boldsymbol{a}, \tau/3, \tau)$ **then**
  Reject and output $\perp$
Accept and output h

Algorithm 2: $\mathsf{Validate}(\boldsymbol{q}, \boldsymbol{a}, \tau_P, \tau)$ i.e., Non-adaptive Statistical Validation

Input: Batch of statistical queries $\boldsymbol{q}$;
  List of candidate answers to queries $\boldsymbol{a}$;
  Candidate accuracy $\tau_P > 0$ and final accuracy $\tau > 0$
Setup: Non-adaptive τ_V-accurate SQ oracle $\mathcal{O}_V$ s.t. $\tau_P + 2\tau_V \leq \tau$
Issue batch of queries $\boldsymbol{q}$ to $\mathcal{O}_V$; let $\hat{a}_j := \mathcal{O}_V(q_j)$ for all $1 \leq j \leq B$
for $j = 1, \ldots, B$ **do**
  if $|\hat{a}_j - a_j| > \tau_V + \tau_P$ **then**
    return False
end
return True

We prove that Protocol 1 is a pvCSV where the verifier's sample complexity scales "non-adaptively" even when an adaptive algorithm is necessary for learning. We start with the following lemma.

Lemma 3.3. *Fix a learning goal $\mathcal{G}$, $\tau > 0$. Suppose $\mathcal{A}$ is a deterministic algorithm that τ-SQ learns $\mathcal{G}$. There is an SQ protocol (P, V), described in Protocol 1, with the following properties:*

(a) (P, V) is a non-interactive SQ protocol that $(\tau/3, \tau/3, 0)$-SQ verifiers $\mathcal{G}$

(b) V uses a non-adaptive SQ oracle, regardless of the SQ adaptivity of $\mathcal{A}$. That is, suppose $\mathcal{A}$ has τ-query complexity (k, B); then,

- *P has $\tau/3$-query complexity $(1, B)$;*
- *V has $\tau/3$-query complexity (k, B)*

(c) V and P run in linear time in the original algorithm. That is, suppose $\mathcal{A}$ runs in time $T_{\mathcal{A}}(\tau)$; then both P and V run in time $O(T_{\mathcal{A}}(\tau))$

Proof. First, we analyze the prover and verifier's query complexities. Observe that P runs a direct simulation of the queries issued by $\mathcal{A}$, supported by $\mathcal{O}_P$. Since $\mathcal{O}_P$ is $\tau/3$-accurate, then P has $\tau/3$-query complexity (k, B). In contrast, V calls $\mathcal{O}_V$ once on all B queries through the Validate subroutine. Since $\mathcal{O}_V$ is $\tau/3$-accurate, V has $\tau/3$-query complexity $(1, B)$. So part (b) holds.

Next, we analyze run time. Observe that both P and V simulate an execution of $\mathcal{A}$. While the oracles $\mathcal{O}_P$ and $\mathcal{O}_V$ are both $\tau/3$-accurate, the simulations of $\mathcal{A}$ can assume τ-accuracy, so the simulations take time $O(T_{\mathcal{A}}(\tau))$. V also runs a statistical valdiation that checks the statistical closeness of the B queries, each of which takes constant time to evaluate in the query model, which is also linear in the run time of $\mathcal{A}$. Thus, part (c) holds.

To complete the lemma, we show that (P, V) is a non-interactive SQ protocol that $(\tau/3, \tau/3, 0)$-SQ verifies $\mathcal{G}$.[9] Observe by construction (P, V) is non-interactive: P sends one message (h, π) to V. Next, we show (P, V) satisfies perfect completenees and universal soundness for $\mathcal{G}$

Completeness: We consider the honest protocol (P, V) where $\mathcal{D}_P = \mathcal{D}_V$. First, we show that V always accepts the honestly generated certificate (h, π). Since (h, π) is honestly generated, V can parse π into queries $\boldsymbol{q}$ and answers $\boldsymbol{a}$. Then it is sufficient to show that π is consistent with an execution of $\mathcal{A}$ that results in the output h. Consider the sequence of queries $\boldsymbol{q} = (q_1, q_2, \ldots, q_B)$ listed in the certificate. We claim this is exactly the same sequence of queries that V will produce when simulating $\mathcal{A}$. This claim follows (formally by induction on the sequence of queries) because $\mathcal{A}$ is deterministic and the verifier's simulation of $\mathcal{A}$ uses the same query responses $\boldsymbol{a} = (\mathcal{O}_P(q_1), \mathcal{O}_P(q_2), \ldots, \mathcal{O}_P(q_B))$ as the prover's execution. By the same argument, the output h of the prover and verifier's execution of $\mathcal{A}$ will be the same (since $\mathcal{A}$ is deterministic and operates on the same input sequence).

Second, we show that π passes the verifier's statistical validation $\mathsf{Validate}(\boldsymbol{q}, \boldsymbol{a}, \tau/3, \tau)$, according to $\mathcal{O}_V$. Recall that $\mathcal{O}_P$ and $\mathcal{O}_V$ are both $\frac{\tau}{3}$-accurate oracles for $\mathcal{D}_P = \mathcal{D}_V$, by assumption. Since the verifier issues the same queries $q_1, q_2, \ldots, q_B$ to $\mathcal{O}_V$, we see that, for each $1 \leq j \leq B$, the comparison looks at the difference between $\mathcal{O}_P(q_j)$ and $\mathcal{O}_V(q_j)$. Both queries are within $\tau/3$ of the true expectation $\mathbf{E}_{x \sim \mathcal{D}}[q_j(x)]$, so by the triangle inequality $|\hat{a}_j - a_j| \leq 2\tau/3$. Therefore, Validate always returns true, and V always accepts the honestly generated (h, π).

[9] The failure probability of the pvCSV will arise when we implement the protocol's oracles using samples.

So it is sufficient to show that $h \in \mathcal{G}(\mathcal{D}_V)$. Recall that h is the output of running $\mathcal{A}$ using a $\tau/3$-accurate oracle for $\mathcal{D}_P$. Because $\mathcal{D}_P = \mathcal{D}_V$ (and $\tau/3 \leq \tau$), this execution is also the result of running $\mathcal{A}$ using *some* τ-accurate oracle for $\mathcal{D}_V$. Thus, by the assumption that $\mathcal{A}$ τ-SQ learns $\mathcal{G}$, then $h \in \mathcal{G}(\mathcal{D}_V)$. So, (P, V) achieves completeness 1.

Universal Soundness: Let $\mathcal{D}_V$ be any distribution, and $\mathcal{O}_V$ be any $\tau/3$-accurate SQ oracle for $\mathcal{D}_V$. Take any cheating prover strategy $\tilde{P}$, and consider the interaction $(\tilde{P}^{\mathcal{O}_V}, V^{\mathcal{O}_V})$. It suffices to argue, as in the above argument for completeness, that if V accepts, then $h \in \mathcal{G}(\mathcal{D}_V)$. If V accepts, then it must be the case that π is consistent with some execution of $\mathcal{A}$ that results in outputting h. Further, $\mathsf{Validate}(\boldsymbol{q}, \boldsymbol{a}, \tau/3, \tau)$ passes, which implies the answers $\boldsymbol{a}$ are within $2\tau/3$ of the responses from the verifier's non-adaptive oracle query $\hat{\boldsymbol{a}} = \mathcal{O}_V(\boldsymbol{q})$. Since $\mathcal{O}_V$ is $\tau/3$-accurate with respect to $\mathcal{D}_V$, by the triangle inequality, $|a_j - \mathbf{E}_{x \sim \mathcal{D}_V}[q_j(x)]| \leq \tau$ for all $1 \leq j \leq B$.

In combination, π gives the verifier a way to simulate $\mathcal{A}$ using τ-accurate answers its queries over $\mathcal{D}_V$. The learning guarantee of $\mathcal{A}$ implies that the returned hypothesis $h \in \mathcal{G}(\mathcal{D}_V)$ must be valid. Thus under interaction with any prover cheater, the verifier V either rejects or accepts and outputs $h \in \mathcal{G}(\mathcal{D}_V)$, and (P, V) satisfies universal soundness 0. □

With this lemma in place, establishing our main theorem of this section is straightforward. To achieve an improved bound on the size of the certificate, we make the following optimization. Notice that our proof of correctness does not actually use the fact that the queries $\boldsymbol{q}$ used by the prover are the same as the queries derived by the verifier's simulation of $\mathcal{A}$; instead, we only need that the answers in the certificate, $\boldsymbol{a}$ are valid for the verifier's queries. Thus, we do not need to include queries in the certificate at all: we need only the certificate be the list of answers, that is, $\pi := \boldsymbol{a}$. While including the names of the queries improves the clarity of the proof system, we derive an equally-valid SQ protocol where $\pi := \boldsymbol{a}$ in the following theorem.

Theorem 3.4 (Formal Statement of Theorem 2)**.** *Fix a learning goal $\mathcal{G}$ and $\tau > 0$. Suppose $\mathcal{A}$ is a deterministic SQ algorithm that τ-SQ learns $\mathcal{G}$, which makes k adaptive queries to a τ-accurate oracle $\mathcal{O}$. For any $\delta > 0$, there exists a pvCSV scheme for $\mathcal{G}$ with failure probability δ achievable in the following complexities.*

- *Honest prover sample complexity:* $m_P \leq O\left(\sqrt{k \cdot \log(k/\delta) \cdot \log(1/\delta)}/\tau^2\right)$
- *Verifier sample complexity:* $m_V \leq O(\log(k/\delta)/\tau^2)$
- *Certificate size:* $|\pi| \leq O(k \cdot \log(1/\tau))$

Proof. Let (P, V) be Protocol 1. By Lemma 3.3, to run a fully-adaptive SQ algorithm using k queries, it suffices to have a $(k, k, \tau/3)$-implementation of $\mathcal{O}_P$ and a $(1, k, \tau/3)$-implementation of $\mathcal{O}_V$. The honest prover and verifier sample complexities follow from Proposition 2.11 and Proposition 2.10, respectively. The original SQ verification protocol contributes $\gamma = 0$ failure probability, so by union bounding over the failure probability of implementing $\mathcal{O}_P$ and $\mathcal{O}_V$, completeness and universal soundness hold with probability at least $1 - \delta$.

Finally, we analyze the size of the certificate π. Without loss of generality, we modify the certificate so that $\pi := \boldsymbol{a}$, and the verifier so that it doesn't check for $\tilde{q}_i = q_i$. As mentioned above, this does

not change the validity of Lemma 3.3. Every answer to a statistical query requires $O(\log(1/\tau))$ bits of space. Since $\pi = \boldsymbol{a} = (a_1, a_2, \ldots, a_B)$, then $|\pi| = O(B \log(1/\tau))$. □

Finally, we remark informally on the time complexities of the honest prover and verifier. Both parties run a direct simulation of the original SQ algorithm $\mathcal{A}$. Additionally, the honest prover must implement the mechanism of [Bla25] for adaptive data analysis to answer the queries to $\mathcal{O}_P$, whereas $\mathcal{O}_V$ may respond with the empirical statistics from its sample. In most settings, the overall running time for both will be dominated by the simulation of $\mathcal{A}$.

4 Delegation of Randomized SQ Learning

In this section, we address delegating SQ learning algorithms that use randomness. Our investigation leads us to develop a number of new definitions and tools for understanding SQ protocols. With these tools in place, we obtain (computationally-sound) pvCSVs, not only for randomized SQ algorithms, but for a large class of SQ protocols.

We start by giving essential definitions, highlighting important properties of randomized SQ algorithms and protocols. Using these definitions, we then show that for a large class of SQ protocols, there exists a reformulation into a Canonical SQ Protocol. This canonical protocol allows for non-adaptive verification of the statistical queries, leading to exponential savings in sample complexity for the verifier, for any original SQ protocol (including randomized algorithms that make adaptive statistical queries). We conclude the section with a construction of pvCSVs for this class, by showing how to apply a Fiat-Shamir transformation to the canonical protocol.

4.1 Definitions for Randomized SQ Learning and SQ Protocols

To begin, we fix a standard, mechanical format for describing randomized SQ algorithms that proceed in *epochs*. In the t-th epoch,

- $\mathcal{A}$ samples randomness $r_t \in \{0,1\}^{d_t}$, for some (efficiently-bounded) $d_t \in \mathbb{N}$
- $\mathcal{A}$ then executes a deterministic SQ algorithm $\mathcal{A}_t$ as a subroutine, with r_t as fixed input, which may issue adaptive statistical queries.

The choice of the t-th subroutine $\mathcal{A}_t$ may be chosen adaptively as a function of prior randomness, as well as any statistical queries issued. In terms of expressivity, formatting randomized algorithms into epochs is not a restriction, as we can simulate any algorithm in this format. But the number of epochs turns out to be a key quantity that affects the complexity of our proof systems.

Definition 4.1 (Epoch Complexity)**.** *Fix $\tau > 0$ and $\ell \in \mathbb{N}$. A randomized SQ algorithm $\mathcal{A}$ has epoch complexity ℓ if for any τ-accuate SQ oracle $\mathcal{O}$, the execution of $\mathcal{A}^{\mathcal{O}}$ can be broken into ℓ epochs.*

Extending the notion of epoch complexity to SQ protocols, we standardize the verifier's execution similarly. In the t-th epoch,

- V samples randomness $r_t \in \{0,1\}^{d_t}$ for some (efficiently-bounded) $d_t \in \mathcal{N}$
- V executes a deterministic, interactive SQ algorithm V_t as a subroutine, with r_t as a fixed input, which may issue adaptive statistical queries and communicate with P.

Again, the choice of the t-th subroutine V_t can be chosen adaptively based on prior randomness, V's internal state, and messages from the prover.

Definition 4.2 (Epoch Complexity of SQ Protocols)**.** *Fix $\tau_V > 0$ and $\ell \in \mathbb{N}$. An SQ verifier V has epoch complexity ℓ if for any prover $\tilde{P}$ and any τ_V-accurate SQ oracle $\mathcal{O}_V$, the execution of V within the interaction $(\tilde{P}^{\mathcal{O}_V}, V^{\mathcal{O}_V})$ can be broken into ℓ epochs.*

Intuitively, formatting randomized algorithms (and SQ verifiers) into epochs allows us to separate "adaptivity" of $\mathcal{A}$ based on fresh randomness and adaptivity based on the results of statistical queries (and prover messages). In the extreme, each epoch could issue a single statistical query, so adaptivity to the randomness and queries are essentially the same. At the other extreme, we might consider algorithms that start with a random initialization, but then proceed deterministically. Such algorithms may still make adaptive SQ queries, but could be implemented in a single epoch. The number of epochs serves as a natural complexity measure of the algorithm's use of its randomness.

Oblivious vs. Public-State SQ Oracles. Recall that we model an SQ oracle as a stateful algorithm, who may respond adversarially (within τ-accuracy) based on the state of the algorithm $\mathcal{A}$. When our algorithms leverage randomness, we need to reason about what knowledge the oracle $\mathcal{O}$ has about the algorithm's internal state—namely, the randomness sampled at the current epoch.

One standard notion of SQ oracle allows the algorithm to maintain private state from the oracle. In this case, the algorithm need not reveal any of its random coins to the oracle. We refer to such oracles as *oblivious* SQ oracles, which follow the earlier notion given in Definition 2.1.

For randomized SQ algorithms, we mainly focus on oracles, who may respond adversarially with full knowledge of the algorithm's randomness. This notion is captured by a *public-state* SQ oracle.

Definition 4.3 (Public-State SQ Oracle)**.** *A public-state SQ oracle is a stateful algorithm $\mathcal{O} : \mathcal{Q} \times \{0,1\}^* \to [0,1]$ that takes as input a query $q \in \mathcal{Q}$ and a string $r \in \{0,1\}^*$ representing random coins, and responds with an evaluation $\mathcal{O}(q,r) \in [0,1]$.*

For $\tau > 0$, the oracle $\mathcal{O}$ is τ-accurate over $\mathcal{D}$ if for any finite, adaptively chosen sequence of $k \in \mathbb{N}$ queries and randomness $(q_1, r_1), (q_2, r_2), \ldots, (q_k, r_k)$, for all $i \in [k]$,

$$\left| \mathcal{O}(q_i, r_i) - \mathop{\mathbf{E}}_{X \sim \mathcal{D}}[q_i(X)] \right| \leq \tau.$$

Concretely, consider a sequence issued by a randomized SQ algorithm $\mathcal{A}$. In the t-th epoch, whenever $\mathcal{A}^t$ issues a query $q \in \mathcal{Q}$, the randomness $r_1 \ldots r_t$ is sent to the public-state oracle $\mathcal{O}(q, r_1 \ldots r_t)$.

Further, a randomized SQ algorithm τ-SQ learns $\mathcal{G}$ with public state, if correctness holds for any τ-accurate public-state SQ oracle $\mathcal{O}$; an SQ protocol (P,V) (τ_P, τ_V, γ)-SQ verifies $\mathcal{G}$ with public verifier state, if completeness and soundness hold for any τ_V-accurate public-state SQ oracle $\mathcal{O}_V$.

In other words, a randomized algorithm equipped with a public-state SQ oracle may, as usual, specify the query $q \in \mathcal{Q}$ of interest, but necessarily reveals the randomness sampled so far. By the standardization into epochs, the randomness captures all of the state of $\mathcal{A}$.

Note that any algorithm/protocol that succeeds with access to a public-state oracle also works with an oblivious oracle, but the converse is not true. Correctness with respect to a public-state oracle is a stronger guarantee: no matter what information is leaked to the oracle through interaction, the algorithm will work. That said, proving correctness may be more challenging, so designing correct algorithms with a public-state oracle may be harder than with an oblivious oracle.

Public-Coin SQ Protocols. As with traditional interactive protocols, we can distinguish SQ protocols based on the verifier's communication of its randomness. Key to our study of non-interactive proof systems for randomized SQ algorithms, we first consider *public-coin* interactive SQ protocols. Defining public-coin protocols when the verifier is equipped with an SQ oracle (which may introduce non-determinism) is nuanced. We consider two alternative definitions.

Definition 4.4 (Public-Coin SQ Protocols)**.** *An SQ protocol (P, V) is a (standard) public-coin SQ protocol if in every round of communication, V samples a uniformly random string r, independent of all prior randomness and messages, and sends r to P. Concretely, for V of epoch complexity ℓ, the protocol consists of ℓ rounds, where in the t-th round, V_t sends its randomness r_t to P.*

Definition 4.5 (Mixed-Message SQ Protocols)**.** *An SQ protocol (P, V), where V has epoch complexity ℓ. (P, V) is a mixed-message (public-coin) SQ protocol if for all $t = 1, \ldots, \ell$, the first message of the t-th epoch is from V_t to P and includes the epoch's randomness r_t.*

In other words, a standard public-coin SQ protocol adopts the formalism that the only messages the verifier sends to the prover are its randomness, whereas a mixed-message SQ protocol must reveal its randomness to the prover, but may also send non-random challenges. Without an SQ oracle, the distinction is moot: in the t-th epoch, V_t is a deterministic algorithm, so the prover can simulate any challenges it would receive and respond accordingly. But with an SQ oracle—which may respond adversarially within its tolerance—$V_t^{\mathcal{O}_V}$ may have non-deterministic behavior, even conditioned on r_t. Thus, in principle, mixed-message SQ protocols could be more expressive than public-coin SQ protocols. Despite this distinction, our main result in Section 4.2 (Protocol 3) implies that (as in standard interactive proofs) any mixed-message SQ protocol can be compiled into a public-coin SQ protocol with the same completeness and soudness guarantees.

Public-Query SQ Protocols. A final consideration in classifying SQ protocols is whether the verifier's queries are kept private or made public to the prover. A public-query protocol reveals its queries to the prover at every round of communication.

Definition 4.6 (Public-Query SQ Protocols)**.** *An SQ protocol (P, V) is a public-query SQ protocol if in every round i of communication, the ith message m_i from V to P includes every statistical query $q_{i_1}, \ldots, q_{i_j}$ issued by V to $\mathcal{O}_V$ since the previous message m_{i-1}.*

Note that, per the discussion above, whether the verifier's queries are public or private is orthogonal to whether the verifier's coins are public or private. That said, Protocol 3 also implies that any

public-coin, private-query protocol can be made public-query. In this sense, we use the term "public-query" to refer to private-coin, public-query SQ protocols. We explore public-query protocols further when we consider the strength and limits of SQ proof systems beyond pvCSVs in Section 5.

4.2 A Canonical Public-Coin SQ Protocol

With the preliminaries on randomized SQ algorithms and SQ protocols in place, we are ready to describe protocols for delegating randomized SQ algorithms. We describe, in Protocol 3, an interactive public-coin SQ protocol for delegating any randomized SQ algorithm that SQ-learns a concept $\mathcal{G}$ with public state. In fact, our protocol is much more general, and can take any mixed-message, private-query SQ protocol that SQ-verifies $\mathcal{G}$ with public verifier state (which include all randomized SQ algorithms with public state) and compile it into a canonical public-coin SQ protocol, where the verifier issues a single non-adaptive batch of statistical queries.

Protocol Description. Protocol 3 works as follows. We start with a mixed-message SQ protocol (P, V) where the verifier may make private statistical queries, and each player is arbitrarily-adaptive in their queries. Note that randomized SQ algorithms are the special case of such protocols, where the algorithm must accept h or output $\perp$ without assistance from any prover. We want to build a public-coin, public-query SQ protocol $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ where the verifier issues a single batch of statistical queries. As in Protocol 1, the verifier will delegate its queries to the prover and check them at the end.

Because (P, V) is mixed-message, we know that the verifier V starts each epoch by sending its randomness to the prover. This random message will be the only message $\mathsf{V}_{\mathsf{can}}$ sends to $\mathsf{P}_{\mathsf{can}}$ per epoch; then, $\mathsf{P}_{\mathsf{can}}$ will be responsible for simulating the remaining interactive execution of that epoch. $\mathsf{P}_{\mathsf{can}}$ will use its own SQ oracle $\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}$ to make any queries. This aspect of the protocol—where the prover simulates the queries of the verifier with knowledge of its randomness—is where we need to leverage the assumption that (P, V) learns $\mathcal{G}$ with public verifier state; that is, even if the oracle (or in this case the prover) can respond adversarially based on the state of the verifier, the protocol is still sound. Once the epoch finishes, $\mathsf{P}_{\mathsf{can}}$ will send the transcript of the execution to $\mathsf{V}_{\mathsf{can}}$, who can update state and move to the next epoch.

At the end, the canonical verifier $\mathsf{V}_{\mathsf{can}}$ checks to ensure that the prover faithfully simulated the execution of P and V. Finally, $\mathsf{V}_{\mathsf{can}}$ runs a statistical check to make sure all of the query values reported by the prover are actually within the required tolerance of the original protocol.

In all, we obtain the following guarantee on our canonical public-coin SQ protocol.

Lemma 4.7 (Formal Statement of Lemma 4)**.** *Fix a learning goal $\mathcal{G}$, $\tau_P, \tau_V > 0$, and $\gamma > 0$, and let $\tau'_P = \min\{\tau_P, \tau_V/3\}$ and $\tau'_V = \tau_V/3$. Suppose (P, V) is a mixed-message, private-query SQ protocol that (τ_P, τ_V, γ)-verifies $\mathcal{G}$ with public verifier state. There is an SQ protocol $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$, described in Protocol 3, with the following properties:*

(a) $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ is a public-coin SQ protocol that $(\tau'_P, \tau'_V, \gamma)$-verifies $\mathcal{G}$

(b) For V of epoch complexity ℓ, $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ has at most ℓ rounds of interaction.

Protocol 3: Canonical SQ Protocol

Setup: Let $\mathcal{G}, \tau_P, \tau_V, \gamma$ and (P, V) be as described in Lemma 4.7.
Let $\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}$ be a $\min\{\tau_P, \tau_V/3\}$-accurate SQ oracle for $\mathcal{D}_P$.
Let $\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}$ be a $\tau_V/3$-accurate SQ oracle for $\mathcal{D}_V$.

Phase 1: *Interactive simulation of SQ protocol* (P, V)
Initialize an empty transcript $\pi_0 = \langle\rangle$
for $i = 1, \ldots, \ell$ **do**
Prover $\mathsf{P}_{\mathsf{can}}$ and **Verifier** $\mathsf{V}_{\mathsf{can}}$ determine V_i for the i-the epoch of V, based on π_{i-1}
Verifier $\mathsf{V}_{\mathsf{can}}$ samples the i-th epoch's randomness r_i and sends to $\mathsf{P}_{\mathsf{can}}$
Prover $\mathsf{P}_{\mathsf{can}}$ updates $\pi_i \leftarrow \langle \pi_{i-1}, r_i \rangle$
$\mathsf{P}_{\mathsf{can}}$ simulates the i-th epoch by executing $(P^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}}, V_i^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}})(\pi_i)$; specifically:
- when V_i issues SQ q, $\mathsf{P}_{\mathsf{can}}$ queries $\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}(q)$ and updates $\pi_i \leftarrow \langle \pi_i, q, \mathcal{O}_{\mathsf{P}_{\mathsf{can}}}(q) \rangle$
- when V_i sends message s to P, $\mathsf{P}_{\mathsf{can}}$ simulates prover $P^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}}(\pi_i)$
- when P sends message m to V, $\mathsf{P}_{\mathsf{can}}$ updates $\pi_i \leftarrow \langle \pi_i, s, m \rangle$

if *simulation of* V *returns* h **then**
$\mathsf{P}_{\mathsf{can}}$ updates $\pi \leftarrow \pi_i$ and sends (h, π) to $\mathsf{V}_{\mathsf{can}}$
$\mathsf{P}_{\mathsf{can}}$ breaks from loop
else if *simulation of* V *ready for next epoch* **then**
$\mathsf{P}_{\mathsf{can}}$ sends updates to transcript π_i to $\mathsf{V}_{\mathsf{can}}$ for next epoch
end

Phase 2: *Verification of interactive transcript*
Verifier $\mathsf{V}_{\mathsf{can}}$**:**
Read and typecheck (h, π). Parse π into $(\boldsymbol{q}_V, \boldsymbol{a}_V, \tilde{\boldsymbol{r}}, \boldsymbol{m}, \boldsymbol{s})$ as follows:
- $\boldsymbol{q}_V := (q_1, q_2, \ldots, q_{B_V})$, the list of verifier SQs issued during simulation of (P, V)
- $\boldsymbol{a}_V := (a_1, a_2, \ldots, a_{B_V})$, the answers to $\boldsymbol{q}_V$, where a_j should equal $\mathcal{O}_P(q_j)$
- $\tilde{\boldsymbol{r}} := (\tilde{r}_1, \ldots, \tilde{r}_\ell)$, the public coins of V
- $\boldsymbol{s} := (s_1, \ldots,)$, the non-random messages of V to P
- $\boldsymbol{m} := (m_1, \ldots,)$, the messages of P to V

Check that π is consistent with (P, V) outputting h; that is,
- Check that $\tilde{\boldsymbol{r}} = r_1, \ldots, r_\ell$
- Simulate V with randomness $\tilde{\boldsymbol{r}}$; use $\boldsymbol{a}_V$ to answer SQs, and $\boldsymbol{m}$ for messages from P
- Check that V returns $h \neq \perp$

if π *is NOT consistent with* (P, V) **then**
Reject and output $\perp$

Phase 3: *Validation of simulated oracle queries*
Verifier $\mathsf{V}_{\mathsf{can}}$**:**
if *NOT* $\mathsf{Validate}(\boldsymbol{q}_V, \boldsymbol{a}_V, \tau_V/3, \tau_V)$ **then**
Reject and output $\perp$
Accept and output h

(c) $\mathsf{V}_{\mathsf{can}}$ *uses a non-adaptive SQ oracle, regardless of the SQ adaptivity in* (P, V)*. That is, suppose* V *has* τ_V*-query complexity* (k_V, B_V) *and* P *has* τ_P*-query complexity* (k_P, B_P)*; then,*

- $\mathsf{V}_{\mathsf{can}}$ *has* τ'_V*-query complexity* $(1, B_V)$*;*
- $\mathsf{P}_{\mathsf{can}}$ *has* τ'_P*-query complexity* $(k_P + k_V, B_P + B_V)$

(d) $\mathsf{V}_{\mathsf{can}}$ *and* $\mathsf{P}_{\mathsf{can}}$ *run in linear time in the original protocol. That is, suppose* V *runs in time* $T_V(\tau_V)$ *and* P *runs in time* $T_P(\tau_P)$*; then*

- $\mathsf{V}_{\mathsf{can}}$ *runs in time* $O(T_V(\tau_V))$
- $\mathsf{P}_{\mathsf{can}}$ *runs in time* $O(T_V(\tau_V) + T_P(\tau_P))$

(e) *For* P *of communication complexity* c_P*,* $\mathsf{P}_{\mathsf{can}}$ *has communication complexity* $O(B_V \log(1/\tau'_P)) + c_P$*.*

Proof. Before we prove that the canonical protocol verifies $\mathcal{G}$, we will analyze the other properties first. First, we analyze round complexity. Observe by construction that each round of communication corresponds to one epoch of V. Since V has epoch complexity ℓ, $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ has round complexity ℓ. So part (b) holds.

Next, we analyze statistical complexity. Observe that $\mathsf{P}_{\mathsf{can}}$ simulates the interaction $(P^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}}, V^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}})$, so $\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}$ must answer at most B_P queries from P and B_V queries from V, and compose the rounds of adaptivity for $k_P + k_V$ rounds. Thus $\mathsf{P}_{\mathsf{can}}$ has τ'_P-query complexity $(k_P + k_V, B_P + B_V)$. Meanwhile, $\mathsf{V}_{\mathsf{can}}$ only checks the statistical validity of the proposed verifier's queries in a single batch to $\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}$, so it has τ'_V-query complexity $(1, B_V)$. Therefore part (c) holds.

Next, we analyze run time. $\mathsf{P}_{\mathsf{can}}$ just simulates the entire interaction $(P^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}}, V^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}})$. While $\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}$ operates at τ'_P accuracy, in order to meet the (τ_P, τ_V, γ)-learning guarantee of the original protocol, $\mathsf{P}_{\mathsf{can}}$ simulates P running at τ_P accuracy and V running at τ_V accuracy. So $\mathsf{P}_{\mathsf{can}}$ runs in time $O(T_V(\tau_V) + T_P(\tau_P))$. Similarly, $\mathsf{V}_{\mathsf{can}}$ simulates V but using transcript π and running at τ_V accuracy. So $\mathsf{V}_{\mathsf{can}}$ runs in time $O(T_V(\tau_V))$, and part (d) holds.

Now, we analyze the $\mathsf{P}_{\mathsf{can}}$'s communication complexity. In the description of Protocol 3, $\mathsf{P}_{\mathsf{can}}$ implicitly sends the entire transcript π of the simulation of (P, V), which comprises of the queries $\boldsymbol{q}_V$ and answers $\boldsymbol{a}_V$ of V, public coins $\tilde{\boldsymbol{r}}$ of V, non-random messages $\boldsymbol{s}$ from V to P, and messages $\boldsymbol{m}$ from P to V. However, strictly speaking $\mathsf{P}_{\mathsf{can}}$ does not need to relay $\boldsymbol{q}_V$, $\tilde{\boldsymbol{r}}$, and $\boldsymbol{s}$. $\mathsf{V}_{\mathsf{can}}$ already has the true public coins $(r_1, \ldots, r_\ell)$, and can simulate V using that, $\boldsymbol{a}_V$ and $\boldsymbol{m}$ to generate the missing $\boldsymbol{q}_V$ and $\boldsymbol{s}$. Therefore, we can simplify $\mathsf{P}_{\mathsf{can}}$'s communication complexity to just $\boldsymbol{a}_V$ and $\boldsymbol{s}$. Then note $|\boldsymbol{a}_V| = O(B_V \log(1/\tau'_P))$. And $|\boldsymbol{s}| = c_P$, the communication complexity of P. So $\mathsf{P}_{\mathsf{can}}$'s communication complexity is $O(B_V \log(1/\tau'_P) + c_P$, and part (e) holds.

To complete the lemma, we show that $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ is a public-coin SQ protocol that $(\tau'_P, \tau'_V, \gamma)$-SQ verifies $\mathcal{G}$. First, observe that $\mathsf{V}_{\mathsf{can}}$ sends all random coins generated for the ℓ rounds of communication, so it is public-coin. As noted in part (c), $\mathsf{P}_{\mathsf{can}}$ employs a τ'_P-accurate SQ oracle, and $\mathsf{V}_{\mathsf{can}}$ employs a τ'_V-accurate SQ oracle. So what remains to be shown is that the canonical transformation preserves $1 - \gamma$ completeness and γ soundness.

Completeness: Assume $\mathcal{D}_P = \mathcal{D}_V$, and consider the honest protocol $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ whose interaction generates (h, π). First, we note that in the honest protocol, π is well-formatted, so the canonical

verifier can parse π into $(\boldsymbol{q}_V, \boldsymbol{a}_V, \tilde{\boldsymbol{r}}, \boldsymbol{s}, \boldsymbol{m})$ and successfully check that π is consistent with an execution of (P, V) that outputs h. That is, V, using randomness $\tilde{\boldsymbol{r}}$ receiving evaluations $\boldsymbol{a}_V$ and messages $\boldsymbol{m}$, will make queries $\boldsymbol{q}_V$, send messages $\boldsymbol{s}$, and output the hypothesis h. This works since $\mathsf{P}_{\mathsf{can}}$ will properly simulate (P, V) and relay the transcript to $\mathsf{V}_{\mathsf{can}}$.

Second, we show that $\boldsymbol{a}$ passes the canonical verifier's statistical validation $\mathsf{Validate}(\boldsymbol{q}_V, \boldsymbol{a}_V, \tau_V/3, \tau_V)$, according to $\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}$. Recall that $\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}$ is τ'_P-accurate for $\mathcal{D}_P$ (which is equal to $\mathcal{D}_V$ by assumption) and $\tau'_P \leq \tau'_V$. Similarly, $\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}$ is τ'_V-accurate for $\mathcal{D}_V$ and $\tau'_V = \tau_V/3$. Thus, by the specification of $\mathsf{Validate}$ (which uses an argument via triangle inequality), statistical validation passes. Additionally since $\tilde{\boldsymbol{r}}$ is generated via $\mathsf{V}_{\mathsf{can}}$'s random coins $(r_1, \ldots, r_\ell)$, in all $\mathsf{V}_{\mathsf{can}}$ will always accept unless $h = \perp$.

Thus, it is sufficient to argue that, under the honest protocol, $h \in \mathcal{G}(\mathcal{D}_V)$ with probability at least $1 - \gamma$. Recall that h is the output of simulating $(P^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}}, V^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}})$ on randomness $\tilde{\boldsymbol{r}}$. Since $\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}$ is a τ'_P-accurate for $\mathcal{D}_P$ and $\tau'_P \leq \tau_P$, then the simulated P has access to a τ_P-accurate oracle for $\mathcal{D}_P$. Similarly, since $\mathcal{D}_P = \mathcal{D}_V$ by assumption, $\tau'_P \leq \tau_V$, and $\mathsf{P}_{\mathsf{can}}$ knows all random coins sent by $\mathsf{V}_{\mathsf{can}}$, then the simulated V has access to a public-state τ_V-accurate oracle for $\mathcal{D}_V$. Since randomness is uniform and generated in epochs according to the canonical verifier, the simulation of $(P^{\mathcal{O}_{\mathsf{P}_{\mathsf{can}}}}, V^{\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}(\tilde{\boldsymbol{r}})})$ is indistinguishable from a true execution of (P, V). Since (P, V) (τ_P, τ_V, γ)-SQ verifies $\mathcal{G}$, then $h \in \mathcal{G}(\mathcal{D}_V)$ with probability at least $1 - \gamma$. Thus, completeness of $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ is $1 - \gamma$.

Universal Soundness: It suffices to show that universal soundness of $\mathsf{V}_{\mathsf{can}}$ reduces to the universal soundness V. Take any cheating prover strategy $\tilde{P}_{\mathsf{can}}$ for $\mathsf{V}_{\mathsf{can}}^{\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}}$. We construct a cheating prover $\tilde{P}$ and oracle $\tilde{O}_V$ for V as follows. We will argue anytime $\tilde{P}_{\mathsf{can}}$ cheat, then $(\tilde{P}, \tilde{O}_V)$ cheats as well. Without loss of generality, let $\tilde{P}_{\mathsf{can}}$ be deterministic. Let $\tilde{P}$ and $\tilde{\mathcal{O}}_V$ each simulate $\tilde{P}_{\mathsf{can}}$, where the randomness of each epoch of V is used as input. If the simulation of $\tilde{P}_{\mathsf{can}}$ produces a valid transcript $\pi = (\boldsymbol{q}_V, \boldsymbol{a}_V, \tilde{\boldsymbol{r}}, \boldsymbol{s}, \boldsymbol{m})$, then $\tilde{\mathcal{O}}_V$ responds as the answers $\boldsymbol{a}_V = (a_1, a_2, \ldots, a_{B_V})$ and $\tilde{P}$ responds as the messages $\boldsymbol{m} = (m_1, m_2, \ldots)$. To ensure that $\tilde{\mathcal{O}}_V$ is a τ_V-accurate oracle for $\mathcal{D}_V$, any answer a which is τ-far from the true expectation for $\mathcal{D}_V$ will be replaced with an arbitrary τ_V-accurate answer for $\mathcal{D}_V$. In the case where $\tilde{P}_{\mathsf{can}}$ does not produce a valid transcript, then $\tilde{P}$ will always reply with a well-formed dummy message and $\tilde{\mathcal{O}}_V$ will always reply with a τ_V-accurate answer.

We now reduce from between $(\tilde{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}}^{\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}})$ and $(\tilde{P}, V^{\tilde{\mathcal{O}}_V})$. Let the random coins of $\mathsf{V}_{\mathsf{can}}$ and V be the same random coins. Let us condition on the event that $(\tilde{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}}^{\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}})$ accepts. Then $\tilde{P}_{\mathsf{can}}$ must have produced a valid transcript $\pi = (\boldsymbol{q}_V, \boldsymbol{a}_V, \tilde{\boldsymbol{r}}, \boldsymbol{s}, \boldsymbol{m})$ and a hypothesis $h \neq \perp$ where π corresponds to a simulation of V that produces h. Further, $\mathsf{Validate}(\boldsymbol{q}_V, \boldsymbol{a}_V, \tau_V/3, \tau_V)$ passed. Then $\boldsymbol{a}_V$ are $2\tau_V/3$-close to the answers from $\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}(\boldsymbol{q}_V)$. Since $\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}$ is $\tau_V/3$-accurate for $\mathcal{D}_V$, then by triangle inequality, $\boldsymbol{a}_V$ are τ_V-accurate for $\mathcal{D}_V$.

Since the random coins are shared between $\mathsf{V}_{\mathsf{can}}$ and V, (h, π), as generated above, is also the output of $\tilde{P}$ and $\tilde{\mathcal{O}}_V$'s simulation of $\tilde{P}_{\mathsf{can}}$. Recall that we are conditioning on the event that the $\mathsf{V}_{\mathsf{can}}$ accepts. Thus, all answers $\boldsymbol{a}_V$ are τ_V-accurate for $\mathcal{D}_V$, and all messages $\boldsymbol{m}$ are well-formed. Thus, $\tilde{P}$ sends messages $\boldsymbol{m}$, and $\tilde{O}_V$ provides answers $\boldsymbol{a}_V$. Therefore, the view of the verifier in $(\tilde{P}, V^{\mathcal{O}_V})$ is indistinguishable from $\mathsf{V}_{\mathsf{can}}$ simulation of V using π. Since $\mathsf{V}_{\mathsf{can}}$ accepted and output h, then V must have accepted and also output h.

Suppose that $(\tilde{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}}^{\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}})$ accepts and outputs $h \notin \mathcal{G}(\mathcal{D}_V)$ with probability γ'. Then by monotonicity, $(\tilde{P}, V^{\tilde{\mathcal{O}}_V})$ accepts and outputs $h \notin \mathcal{G}(\mathcal{D}_V)$ with probability at least γ'. But $(\tilde{P}, V^{\tilde{\mathcal{O}}_V})$ accepts

and outputs $h \notin \mathcal{G}(\mathcal{D}_V)$ with probability at most γ by universal soundness of (P, V). Therefore, $\gamma' \leq \gamma$. Since this holds over all $\mathcal{D}_V$ and $\tau_V/3$-accurate oracles $\mathcal{O}_V$ for $\mathcal{D}_V$, then $(\tilde{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}}^{\mathcal{O}_{\mathsf{V}_{\mathsf{can}}}})$ accepts and outputs $h \notin \mathcal{G}(\mathcal{D}_V)$ with probability at most γ. So $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ satisfies universal soundness γ. □

4.3 pvCSVs from the Canonical Protocol and Fiat-Shamir

Next, we construct pvCSVs for SQ computations that use randomness. Specifically, we compile the canonical Protocol 3 from above into non-interactive SQ protocols using a Fiat-Shamir transformation [FS86]. We prove soundness of the transformation in the Random Oracle Model (ROM), which allows us to replace the public coins of the verifier with non-interactive calls to the random oracle. We begin with preliminaries defining correctness and soundness in the ROM. Then, we describe our Fiat-Shamir transformation over canonical SQ protocols in Protocol 4.

pvCSVs in the ROM. In the Random Oracle Model [BR93], both the prover and verifier have query access to a shared random function called the random oracle $f : \{0,1\}^* \to \{0,1\}^m$ for some output size $m \in \mathbb{N}$. Completeness and soundness take probabilities over the sampling of the random oracle which is denoted $f \leftarrow \mathcal{U}$. Furthermore, we restrict the cheating prover to at most $t \in \mathbb{N}$ queries of the random oracle, called the random oracle budget, typically taken to be polynomial in some security parameter. We adapt Definition 3.2 of pvCSVs to define computationally-sound pvCSVs in the ROM.

Definition 4.8 (pvCSV in the ROM)**.** *Let $\mathcal{G}$, $\tau_P, \tau_V > 0$, $\gamma_{\mathsf{csv}} > 0$, and $(\mathsf{P}_{\mathsf{csv}}, \mathsf{V}_{\mathsf{csv}})$ be defined in the setup of Definition 3.2. The pvCSV prover $\mathsf{P}_{\mathsf{csv}}$ and verifier $\mathsf{V}_{\mathsf{csv}}$ are given query-access to a random oracle f. The pvCSV $(\tau_P, \tau_V, \gamma, \gamma')$-certifies $\mathcal{G}$ in the ROM if the following guarantees hold:*

- *γ-Completeness in the ROM: if $\mathcal{D}_P = \mathcal{D}_V$, then for any verifier distribution $\mathcal{D}_V$, τ_V-accurate oracle $\mathcal{O}_V$ for $\mathcal{D}_V$, τ_P-accurate oracle $\mathcal{O}_P$ for $\mathcal{D}_V$, over sampling of the random oracle f, random coins of $\mathsf{P}_{\mathsf{csv}}$ and $\mathsf{V}_{\mathsf{csv}}$, the honest prover can generate a hypothesis-certificate pair $(h, \pi) \leftarrow \mathsf{P}_{\mathsf{csv}}^{\mathcal{O}_P, f}$ such that $\mathsf{V}_{\mathsf{csv}}^{\mathcal{O}_V, f}$ accepts and $h \in \mathcal{G}(\mathcal{D}_V)$ with probability at least $1 - \gamma$. That is,*

$$\forall \mathcal{D}_V, \mathcal{O}_V, \mathcal{O}_P.\ \mathbf{Pr}\left[\begin{array}{l|l} \mathsf{V}_{\mathsf{csv}}^{\mathcal{O}_V, f}(h, \pi) = 1 & f \leftarrow \mathcal{U}, \\ \wedge\ h \in \mathcal{G}(\mathcal{D}_V) & \text{random coins of } \mathsf{P}_{\mathsf{csv}} \text{ and } \mathsf{V}_{\mathsf{csv}}, \\ & (h, \pi) \leftarrow \mathsf{P}_{\mathsf{csv}}^{\mathcal{O}_P, f} \end{array}\right] \geq 1 - \gamma$$

- *γ'-Universal Soundness in the ROM: for any verifier distribution $\mathcal{D}_V$ and τ_V-accurate oracle $\mathcal{O}_V$ for $\mathcal{D}_V$, for any random oracle budget $t \in \mathbb{N}$, for any t-query prover strategy $\tilde{P}_{\mathsf{csv}}$, for $(h, \pi) \leftarrow \tilde{P}_{\mathsf{csv}}^{\mathcal{O}_V, f}$, then $V^{\mathcal{O}_V}$ rejects or $h \in \mathcal{G}(\mathcal{D}_V)$ with probability at least $1 - \gamma'(t)$ over sampling of the random oracle f, random coins of $\mathsf{V}_{\mathsf{csv}}$. That is,*

$$\forall \mathcal{D}_V, \mathcal{O}_V, \tilde{P}_{\mathsf{csv}}.\ \mathbf{Pr}\left[\begin{array}{l|l} \mathsf{V}_{\mathsf{csv}}^{\mathcal{O}_V, f}(h, \pi) = 0 & f \leftarrow \mathcal{U}, \\ \vee\ h \in \mathcal{G}(\mathcal{D}_V) & \text{random coins of } V, \\ & (h, \pi) \leftarrow \tilde{P}_{\mathsf{csv}}^{\mathcal{O}_V, f} \end{array}\right] \geq 1 - \gamma'(t)$$

We denote ROM soundness against t-query provers as $\gamma'(t)$ where γ' is a function and $t \in \mathbb{N}$ is the random oracle query budget. Additionally, since soundness is parameterized whereas completeness is not, we split completeness and soundness error into (γ, γ').

Applying Fiat-Shamir to the Canonical Protocol. We transform the Canonical SQ protocol $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ into a pvCSV in the ROM by applying the Fiat-Shamir transformation to the initial phase of public-coin interaction. As is standard in Fiat-Shamir, the canonical verifier's randomness in the i-th round is the query of the random oracle with the partial transcript of interaction up to round i. We describe the resulting protocol $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ in Protocol 4. The honest prover P_{FS} simulates the interaction $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ where $\mathsf{V}_{\mathsf{can}}$'s randomness is sampled via the random oracle f. Then, the verifier V_{FS} runs the second and third phase of the canonical protocol as before, checking the correctness of the computational simulation and the statistical validity of the relevant SQs via a batch evaluation of the queries of $\mathsf{V}_{\mathsf{can}}$.

The correctness of the resulting pvCSV follows from the state restoration argument of [BSCS16], with some subtleties that arise in the SQ protocol setting. The structure of the Canonical protocol simplifies the analysis considerably: the protocol naturally divides into a phase of public-coin interaction, followed by (computational and statistical) verification of the transcript. Completeness of the protocol follows from completeness of Protocol 3, since the honest prover can execute the same sequence of queries and computations. Thus, we focus on establishing soundness.

There are two key aspects of Protocol 4 that require us to be careful in establishing soundness of the transformation. First, in our setting, the cheating prover has considerable powers related to the statistical learning problem, with full knowledge of the underlying distribution $\mathcal{D}$ and access to the verifier's SQ oracle. But, importantly, the cheating prover in the Fiat-Shamir protocol and Canonical protocol are afforded the same powers. The approach of establishing Fiat-Shamir soundness via state restoration is a black-box reduction, so we can apply the same argument even in our setting where the provers have non-standard computational and statistical powers. Second, our application of Fiat-Shamir is used to generate a legitimate prover-verifier transcript, which produces a hypothesis h, rather than directly certifying a known predicate (e.g., certifying a given $h \in \mathcal{G}(\mathcal{D}_V)$). That said, once we have a sound transcript, Phase 2 and Phase 3 of the original protocol allow us to validate that the h is actually good for the verifier's distribution.

Specifically, we invoke the following guarantee about the Fiat-Shamir transformation applied to public-coin protocols.

Theorem 4.9 (Corollary of Lemma 13.2.7 and Theorem 14.3.1 of [CY24])**.** *Let (P, V) be a public-coin interactive protocol with round complexity ℓ and soundness γ. Let $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ be the Fiat-Shamir transformed protocol, and suppose it has γ_{FS} soundness in the ROM. There is a black-box reduction that establishes the following upper bound on soundness of $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ in terms of soundness of (P, V), round complexity ℓ, and random oracle budget $t \in \mathbb{N}$.*

$$\gamma_{\mathsf{FS}}(t) \leq \binom{t+\ell}{\ell}\gamma$$

Specifically, the reduction goes through a state restoration game and demonstrates how, given a cheating prover for the Fiat-Shamir proof system $\tilde{\mathsf{P}}_{\mathsf{FS}}$, there exists a cheating prover $\tilde{P}$ for the

Protocol 4: Non-Interactive SQ Protocol in Random Oracle Model

Setup: Let $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ be a canonical SQ protocol that τ-SQ verifies $\mathcal{G}$, where

- P_{FS} accesses a τ-accurate SQ Oracle $\mathcal{O}_P$ for $\mathcal{D}_P$.
- V_{FS} accesses a τ-accurate SQ Oracle $\mathcal{O}_V$ for $\mathcal{D}_V$.
- ℓ is the round complexity of $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$.
- Both P_{FS} and V_{FS} have access to a random oracle f.

Prover $\mathsf{P}_{\mathsf{FS}}^{\mathcal{O}_P, f}$:

Initialize an empty transcript $\pi_0 = \langle\rangle$
Simulate the interaction $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ as follows:
for $i = 1, \ldots, \ell$ **do**
  $\mathsf{P}_{\mathsf{can}}^{\mathcal{O}_P}$ sends $\mathsf{V}_{\mathsf{can}}$ a message m_i
  Generate $\mathsf{V}_{\mathsf{can}}$'s randomness at the i-th round as $r_i := f(\langle m_1, \ldots, m_i\rangle)$
  $\mathsf{V}_{\mathsf{can}}$ sends r_i to $\mathsf{P}_{\mathsf{can}}$
  Update $\pi_i := \langle \pi_{i-1}, m_i, r_i\rangle$
end
Let $\pi := \pi_\ell$
Send π to **Verifier** V_{FS}.

Verifier $\mathsf{V}_{\mathsf{FS}}^{\mathcal{O}_V, f}$:

Parse the transcript π as

- $\boldsymbol{m} = (m_1, m_2, \ldots, m_\ell)$ purported messages from $\mathsf{P}_{\mathsf{can}}$
- $\boldsymbol{r} = (r_1, r_2, \ldots, r_\ell)$ purported generated randomness for $\mathsf{V}_{\mathsf{can}}$

for $i = 1, \ldots, \ell$ **do**
  if $r_i \neq f(\langle m_1, \ldots, m_i\rangle)$ **then**
    Reject and output $\perp$
end
Execute **Phase 2** and **Phase 3** of $\mathsf{V}_{\mathsf{can}}$
if *either phase fails* **then**
  Reject and output $\perp$.
Accept and output h.

original interactive protocol that makes calls to $\tilde{\mathsf{P}}_{\mathsf{FS}}$, at a $\binom{t+\ell}{\ell}$-factor loss in success probability. With this fact, we establish the correctness of Protocol 4.

Lemma 4.10 (Fiat-Shamir for Canonical Protocol)**.** *Fix $\mathcal{G}$, $\tau_P, \tau_V > 0$, $\gamma_{\mathsf{can}} > 0$, and canonical SQ protocol $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ that $(\tau_P, \tau_V, \gamma_{\mathsf{can}})$-verifies $\mathcal{G}$ with round complexity ℓ. Then the Fiat- Shamir transformed protocol $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$, described in Protocol 4, is a pvCSV that $(\tau_P, \tau_V, \gamma_{\mathsf{can}}, \binom{t+\ell}{\ell}\gamma_{\mathsf{can}})$-certifies $\mathcal{G}$ in the ROM where $t \in \mathbb{N}$ is the random oracle budget.*

Proof sketch. Completeness is immediate, by the fact that the random oracle calls are identically distributed to the challenges sent by the public-coin verifier. Thus, the honest prover generates a transcript from the same distribution as the honest execution of Protocol 3.

Soundness follows by Theorem 4.9 applied to Phase 1 of Protocol 3. In particular, the black-box reduction allows us to convert any cheating prover for Protocol 4 $\tilde{\mathsf{P}}_{\mathsf{FS}}$ into a cheating prover for

Protocol 3 $\tilde{\mathsf{P}}_{\mathsf{can}}$. While the provers are afforded non-standard SQ oracles and knowledge of the distribution, the Canonical interactive prover $\tilde{\mathsf{P}}_{\mathsf{can}}$ has the same oracle access as the Fiat-Shamir prover $\tilde{\mathsf{P}}_{\mathsf{FS}}$, so $\tilde{\mathsf{P}}_{\mathsf{can}}$ can implement the black-box calls to $\tilde{\mathsf{P}}_{\mathsf{FS}}$ in the reduction. Thus, converting the interactive SQ protocol into a non-interactive protocol is sound up to the $\gamma_{\mathsf{FS}}(t) \leq \binom{t+\ell}{\ell}\gamma_{\mathsf{can}}$ loss as stated in Theorem 4.9. In particular, the Fiat-Shamir protocol generates a legitimate transcript of the original Canonical protocol, with all but $\gamma_{\mathsf{FS}}(t)$ soundness error.

Finally, we argue that in the SQ model, the verifier's additional checks establish that $h \in \mathcal{G}(\mathcal{D}_V)$ or result in rejection. After the non-interactive simulation of the interactive Phase 1, the verifier additionally executes Phase 2 and Phase 3 of Protocol 3. Phase 2 certifies that the transcript is a legitimate execution of the SQ protocol (based on the responses to statistical queries) and Phase 3 validates that the query responses are statistically correct. In the SQ model, these checks contribute zero additional soundness error. Thus, the upper bound on soundness holds as claimed. □

pvCSVs for all mixed-message protocols with public-verifier state. We conclude with a statement of the overall pvCSV guarantee established in this section within the ROM. In Lemma 4.7, we argue that for the class of mixed-message, private-query SQ protocols with public verifier state can be transformed into a Canonical public-coin SQ protocol. Then, in Lemma 4.10, we apply the Fiat-Shamir transform to turn any Canonical protocol into a pvCSV in the ROM. Chaining these lemmas together, we obtain the following theorem.

Theorem 4.11. *Fix $\mathcal{G}$, $\tau_P, \tau_V > 0$, $\gamma > 0$, and a mixed-message, private-query SQ protocol (P, V) with public verifier state. Let $\tau'_P = \min\{\tau_P, \tau_V/3\}$ and $\tau'_V = \tau_V/3$. Suppose (P, V) (τ_P, τ_V, γ)-verifies $\mathcal{G}$. Applying the canonical transformation described in Protocol 3, and then the Fiat-Shamir transformation, described in Protocol 4, yields a pvCSV $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ that certifies $\mathcal{G}$ with the following characteristics:*

(a) For V with epoch complexity ℓ, $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ $(\tau'_P, \tau'_V, \gamma, \binom{t+\ell}{\ell}\gamma)$-certifies $\mathcal{G}$ in the ROM.

(b) V_{FS} uses a non-adaptive SQ oracle, regardless of the SQ adaptivity in (P, V). That is, suppose V has τ_V-query complexity (k_V, B_V) and P has τ_P-query complexity (k_P, B_P); then,

- *V_{FS} has τ'_V-query complexity $(1, B_V)$;*
- *P_{FS} has τ'_P-query complexity $(k_P + k_V, B_P + B_V)$*

(c) V_{FS} and P_{FS} run in linear time in the original protocol. That is, suppose V runs in time $T_V(\tau_V)$ and P runs in time $T_P(\tau_P)$; then

- *$\mathsf{V}_{\mathsf{can}}$ runs in time $O(T_V(\tau_V))$*
- *$\mathsf{P}_{\mathsf{can}}$ runs in time $O(T_V(\tau_V) + T_P(\tau_P))$*

(d) For P with communication complexity c_P, the certificate size $|\pi| = O(B_V \log(1/\tau'_P)) + c_P$.

Proof. Let $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ be the intermediate, canonical SQ protocol. By Lemma 4.7, $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ $(\tau'_P, \tau'_V, \gamma)$-verifies $\mathcal{G}$. Since V has epoch complexity ℓ, $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ has round complexity ℓ. Therefore, by Lemma 4.10, $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ $(\tau'_P, \tau'_V, \gamma, \binom{t+\ell}{\ell}\gamma)$-certifies $\mathcal{G}$ in the ROM. So part (a) holds.

Observe that the Fiat-Shamir transformation from $(\mathsf{P}_{\mathsf{can}}, \mathsf{V}_{\mathsf{can}})$ to $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ preserves many properties including the precision, query complexity, and runtime of the prover and verifier, (at least while assuming unit cost for evaluating the random oracle). Therefore, parts (b) and (c) follow directly from Lemma 4.7.

Let c_P be the communication complexity of P. By Lemma 4.7, $\mathsf{P}_{\mathsf{can}}$'s communication complexity is $O(B_V \log(1/\tau'_P)) + c_P$. After applying the Fiat-Shamir transformation, the communication complexity is the canonical prover's communication complexity. So, part (d) holds. □

By application of known implementations for adaptive and non-adaptive SQ oracles in Proposition 2.11 and Proposition 2.10, we have the following.

Corollary 4.12. *Let $\mathcal{G}$, τ_P, τ_V, γ, (P, V), pvCSV $(\mathsf{P}_{\mathsf{FS}}, \mathsf{V}_{\mathsf{FS}})$ be defined as in the above theorem. Let $\tau'_P = \min\{\tau_P, \tau_V/3\}$. Suppose that both P and V make fully-adaptive queries, that is, $k_P = B_P$ and $k_V = B_V$. Then for all failure probability $\delta > 0$, the pvCSV has sample complexity $(m_{\mathsf{P}_{\mathsf{FS}}}, m_{\mathsf{V}_{\mathsf{FS}}})$ is as follows:*

$$m_{\mathsf{P}_{\mathsf{FS}}} = O\left(\frac{\sqrt{(k_V + k_P) \cdot \log((k_V + k_P)/\delta) \cdot \log(1/\delta)}}{\tau'^2_P}\right) \qquad m_{\mathsf{V}_{\mathsf{FS}}} = O\left(\frac{\log(k_V/\delta)}{\tau^2_V}\right)$$

5 Beyond pvCSVs: Strengths and Limits of SQ Protocols

In this section, we consider the power and limitations of SQ protocols. First, we reiterate that, statistically, SQ protocols are very powerful. In even more generic settings than our pvCSV constructions, SQ algorithms/protocols can be delegated via interactive SQ protocol such that the verifier's sample complexity scales logarithmically in the number of queries. Then, we show that SQ protocols—despite their sample efficiency—do not generically provide a computationally efficient verification scheme. In particular, by a lower bound of [MS23] on the sample complexity required to PAC Verify certain VC classes, we show a subexponential lower bound on the query complexity of SQ verification for the same class.

Interactive Non-Adaptive SQ Verification. One of the key limitations in our construction of pvCSVs is the reliance on correctness under a public-state SQ oracle. We show that it is possible to achieve non-adaptive statistical verification of SQ algorithms that are only correct under an oblivious SQ oracle, albeit with interaction. This result is analogous to a result showed in the recent journal version of [MS23]. Given any SQ algorithm, the verifier simply executes the algorithm using the prover as its oracle. The prover responds interactively to each query. And then at the end, the verifier checks the answers of the prover non-adaptively.

In fact, this simple idea also establishes that a large class of SQ protocols—even more general than those covered by Lemma 4.7—can be verified using non-adaptive statistical complexity. Specifically, for any public-query (private-coin) SQ protocol that verifies $\mathcal{G}$ with public verifier state, there is an implementation of the protocol that only requires non-adaptive verifier sample complexity.

Proposition 5.1. *Fix a learning goal $\mathcal{G}$, $\tau_P, \tau_V > 0$, and $\gamma > 0$, and let $\tau'_P = \min\{\tau_P, \tau_V/3\}$ and $\tau'_V = \tau_V/3$. Suppose (P, V) is a public-query SQ protocol that (τ_P, τ_V, γ)-verifies $\mathcal{G}$ with public*

verifier state, where P has τ_P-query complexity (k_P, B_P) and V has τ_V-query complexity (k_V, B_V). There is an SQ protocol (P', V') that $(\tau'_P, \tau'_V, \gamma)$-verifies $\mathcal{G}$ with public verifier state, where P' has τ'_P-query complexity $(k_P + k_B, B_P + B_V)$ and V' has non-adaptive τ'_V-query complexity $(1, B_V)$.

Proposition 5.1 follows, again, by asking the prover to make the verifier's statistical queries. The completeness, soundness, and resulting bounds are analogous to those established by Lemma 4.7.

Query Lower Bound for SQ Verification of a VC Class. We show that the statistical upper bound we achieve for the SQ verifier actually implies a computational lower bound for SQ protocols. Specifically, there exists a hypothesis class of VC dimension d that cannot be SQ verified, even using the most general interactive SQ protocols, using polynomially many queries in d. Our lower bound piggybacks off of the lower bound proved in [MS23] for PAC Verification.[10]

Theorem (Restatement of Theorem 2.1 of [MS23])**.** *Fix $\varepsilon > 0$, $\delta = 1/3$, and let (P, V) be an interactive proof system for learning. For any hypothesis class $\mathcal{H}$ of VC Dimension d, if (P, V) PAC verifies $\mathcal{H}$ (for all distributions $\mathcal{D}$) with accuracy ε and failure probability δ, then the verifier V must use at least $m_V \geq \Omega(\sqrt{d}/\varepsilon^2)$ i.i.d. samples from $\mathcal{D}$.*

Stringing this lower bound on the sample complexity of PAC verifying a VC class with our upper bound on the sample complexity of SQ verification, we immediately get a query complexity lower bound for SQ verification. Taking our upper bound of $\log(k)/\tau^2$ from Proposition 5.1, in terms of the number of statistical queries k, and their lower bound of $\sqrt{d}/\varepsilon^2$ in terms of the VC dimension d, we obtain the following corollary.

Corollary 5.2. *Fix $\gamma = 1/3$ and fix the verifier tolerance $\tau_V = \Omega(\varepsilon) > 0$ in terms of the agnostic learning accuracy ε. For every hypothesis class $\mathcal{H}$ of VC dimension d, and for any $\tau_P > 0$, any public-query SQ protocol that (τ_P, τ_V, γ)-SQ verifies ε-Agnostic PAC learning of $\mathcal{H}$ with public verifier state requires the verifier to make $k = 2^{\Omega(\sqrt{d})}$ statistical queries.*

That is, even for the strongest model of SQ protocol we consider—interactive, public-query SQ protocols that use a public-state SQ oracle—when the tolerance of the SQ oracle is fixed $\tau_V \approx \varepsilon$, verification may be computationally inefficient, despite statistical efficiency.

6 Differential Privacy and SQ Verification

In many statistical analyses, Differential Privacy (DP) [DMNS06] is a desirable property to satisfy to maintain the privacy of individuals' data within a database.[11] DP protects individuals' privacy by requiring stability in a randomized algorithm's behavior on neighboring databases D, D' that differ on a single element.

[10] Informally, PAC Verification is the problem of delegation of learning, for the specific learning goal of Agnostic PAC learning. We refer the unfamiliar reader to [GRSY21].

[11] DP is the gold standard notion for privacy-protections in statistical analyses. We refer the unfamiliar reader to introductory materials on DP, including [DR14, Vad17, Kam20]. For the sake of presentation, we omit background on the Exponential Mechanism of [MT07].

Algorithm 5: $\mathsf{DPValidate}(\boldsymbol{v}, \boldsymbol{a}, \tau_P, \tau)$ i.e., Differentially-Private Statistical Validation

Input: Batch of statistical queries $\boldsymbol{q}$;
List of candidate answers to queries $\boldsymbol{a}$;
Candidate accuracy $\tau_P > 0$ and final accuracy $\tau > 0$, where $\tau_P \leq \tau/3$

Setup: m_V i.i.d. samples drawn from $\mathcal{D}_V$; $x_1, \ldots, x_{m_V}$
Sensitiviy of empirical queries $\Delta = 1/m_V$

for $j = 1, \ldots, B$ **do**
  Define $\hat{a}_j = \frac{1}{m_V} \sum_{i=1}^{m_V} q_j(x_i)$
  Define $\nu_j = |\hat{a}_j - a_j|$
end
Use the Exponential Mechanism to sample a noisy maximum ν^* over $\{\nu_j\}$ according to:

$$\mathbf{Pr}[\nu^* = \nu_j] \propto \exp\left(\frac{\varepsilon \cdot \nu_j}{2\Delta}\right)$$

if $\nu^* > \tau/2$ **then**
  return False
return True

Definition 6.1 (Differential Privacy [DMNS06])**.** *Fix a domain $\mathcal{X}$ and range $\mathcal{R}$, and $\varepsilon, \delta > 0$. A randomized algorithm $M : \mathcal{X}^* \to \mathcal{R}$ is (ε, δ)-differentially private if for all neighboring databases $D, D' \in \mathcal{X}^*$ and for all measurable subsets $S \subseteq \mathcal{R}$,*

$$\mathbf{Pr}[M(D) \in S] \leq e^{\varepsilon} \cdot \mathbf{Pr}[M(D') \in S] + \delta.$$

DP provides rigorous protections to individuals, but comes at a cost. Even for non-adaptive statistical queries, in high-dimensional settings ($d \geq k$), answering k queries requires sample complexity scaling polynomially in k. Concretely, to answer k queries with τ tolerance under pure $(\varepsilon, 0)$-DP requires $\Omega(k/\tau\varepsilon)$ i.i.d. samples from $\mathcal{D}$ [HT10]; under approximate (ε, δ)-DP requires $\Omega(\sqrt{k \cdot \log(1/\delta)}/\tau\varepsilon)$ [SU16]. Both of these results are tight (up to poly-logarithmic factors); in fact, the upper bound on answering adaptive statistical queries is tightly connected to the upper bound on answering queries under approximate DP.

Certifying the results of a statistical analysis, while maintaining DP with respect to the verifier's samples, offers a potential for significant savings. We show, generically, that the verifiers for our pvCSVs (in fact, for all of our SQ protocols) can be implemented under DP using essentially the same non-adaptive sample complexity.

Proposition 6.2. *There exists an $(\varepsilon, 0)$-DP implementation of the non-adaptive statistical validation, given in Algorithm 5, with failure probability β that uses m_V i.i.d. samples from $\mathcal{D}_V$, where for any number of adaptive queries k,*

$$m_V \leq O\left(\frac{\log(k/\beta)}{\tau^2} + \frac{\log(k/\beta)}{\tau\varepsilon}\right).$$

Proof sketch. The proposition follows from a standard application of the Exponential Mechanism [MT07]. The original non-adaptive statistical validation step evaluates each query on the samples

from $\mathcal{D}_V$.[12] Namely, the original verifier computes $\hat{a}_1, \ldots, \hat{a}_k$ where $\hat{a}_j = \frac{1}{m_V} \sum_{i=1}^{m_V} q_j(x_i)$. Instead, the new verifier checks the (noisy) maximum difference between the statistics reported by the prover and the empirical statistics on their samples, $|a_j - \hat{a}_j|$.

To compute the new verifier's sample complexity, we need to reason about the accuracy of the empirical statistics $\hat{a}_1, \ldots, \hat{a}_k$, as well as the accuracy of the exponential mechanism. For target tolerance τ and prover tolerance $\tau/3$ (as in our protocols), we will insist that each of these components achieves $\tau/16$ additive error with all but $\beta/2$ probability each.

First, the accuracy of the empirical statistics: per Proposition 2.10, m_V can scale as $O(\log(k/\beta)/\tau^2)$. Then, the accuracy of the release of the maximum difference $\max_{1 \le j \le k} |a_j - \hat{a}_j|$: per [MT07], with all but $\beta/2$ probability, the exponential mechanism achieves error α where

$$\alpha \le \frac{2\Delta}{\varepsilon} \cdot \log(2k/\beta)$$

where $\Delta = 1/m_V$ is the sensitivity of the release from a database of m_V samples. Thus, the exponential mechanism guarantees $\tau/16$-accuracy to the empirical statistics for some $m_V \le O(\log(k/\beta)/\tau\varepsilon)$. With these accuracies fixed with all but β total failure probability, we can define the new verifier and analyze Completeness and Soundness.

For a sequence of k fully-adaptive SQs, Algorithm 5 runs the exponential mechanism to release the noisy maximum difference $\max_{1 \le j \le k} |a_j - \hat{a}_j|$. If the difference is less than $\tau/2$, it accepts; otherwise, it rejects.

Completeness: Per our earlier protocols, we assume the honest prover reports each a_j within $\tau_P = \tau/3$ of the true query value. The verifier's empirical statistics $\hat{a}_j$ are within $\tau/16$ of the true query value, and their difference is released with $\tau/16$-accuracy, so in sum, the reported maximum difference will be strictly less than $\tau/2$, with all but β probability.

Soundness: Suppose there is some statistic a_j that was reported with more than τ additive error from the true value, so the SQ guarantee is violated. Again, the verifier's accuracy ensures that the difference $|a_j - \hat{a}_j|$ is at least $15\tau/16$, which is released with at most $\tau/16$ additive error. So, with all but β probability, the reported maximum difference is strictly more than $\tau/2$. □

[12] This is the simplest implementation of Algorithm 2 given i.i.d. samples.

Acknowledgements

The authors thank Noah Stephens-Davidowitz for significant conversations at the start of this work and feedback throughout the project. We also thank Robert Kleinberg, Jonathan Shafer, and Nick Spooner for helpful discussions.

References

[Bab85] László Babai. Trading group theory for randomness. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 421–429, 1985.

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 106–115. IEEE, 2001.

[BGKW24] Zoë Ruha Bell, Shafi Goldwasser, Michael P Kim, and Jean-Luc Watson. Certifying private probabilistic mechanisms. In *Annual International Cryptology Conference*, pages 348–386. Springer, 2024.

[Bla25] Guy Blanc. Subsampling suffices for adaptive data analysis. *Journal of the ACM*, 72(1):1–45, 2025.

[BNS+16] Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 1046–1059, 2016.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BSCS16] Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In *Theory of Cryptography Conference*, pages 31–60. Springer, 2016.

[BTFM+25] Zoë Ruha Bell, Anvith Thudi, Olive Franzese-McLaughlin, Nicolas Papernot, and Shafi Goldwasser. Efficient public verification of private ml via regularization. *arXiv preprint arXiv:2512.04008*, 2025.

[CG18] Alessandro Chiesa and Tom Gur. Proofs of proximity for distribution testing. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, pages 53–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.

[CY24] Alessandro Chiesa and Eylon Yogev. *Building Cryptographic Proofs from Hash Functions*. 2024.

[DFH+15a] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toni Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. *Advances in neural information processing systems*, 28, 2015.

[DFH+15b] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, 2015.

[DFH+15c] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Leon Roth. Preserving statistical validity in adaptive data analysis. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 117–126, 2015.

[DK22] Yuval Dagan and Gil Kur. A bounded-noise mechanism for differential privacy. In *Conference on Learning Theory*, pages 625–661. PMLR, 2022.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

[DR14] Cynthia Dwork and Aaron Roth. *The Algorithmic Foundations of Differential Privacy*, volume 9. Now Publishers Inc., 2014.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the theory and application of cryptographic techniques*, pages 186–194. Springer, 1986.

[FS18] Vitaly Feldman and Thomas Steinke. Calibrating noise to variance in adaptive data analysis. In *Conference On Learning Theory*, pages 535–544. PMLR, 2018.

[GJK+24] Tom Gur, Mohammad Mahdi Jahanara, Mohammad Mahdi Khodabandeh, Ninad Rajgopal, Bahar Salamatian, and Igor Shinkar. On the power of interactive proofs for learning. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 1063–1070, 2024.

[GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in) security of the fiat-shamir paradigm. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 102–113. IEEE, 2003.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: Interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)*, pages 113–122, Victoria, British Columbia, Canada, 2008. ACM.

[GL89] Oded Goldreich and Leonid Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pages 25–32. ACM, 1989.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (STOC '85)*, pages 291–304, Providence, Rhode Island, 1985. ACM.

[GRSY21] Shafi Goldwasser, Guy N Rothblum, Jonathan Shafer, and Amir Yehudayoff. Interactive proofs for verifying machine learning. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, pages 41–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021.

[Hau92] David Haussler. Decision-theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992.

[HR10] Moritz Hardt and Guy N Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st annual symposium on foundations of computer science*, pages 61–70. IEEE, 2010.

[HR22] Tal Herman and Guy N Rothblum. Verifying the unseen: interactive proofs for label-invariant distribution properties. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1208–1219, 2022.

[HR23] Tal Herman and Guy Rothblum. Doubley-efficient interactive proofs for distribution properties. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 743–751. IEEE, 2023.

[HR24a] Tal Herman and Guy Rothblum. How to verify any (reasonable) distribution property: Computationally sound argument systems for distributions. *arXiv preprint arXiv:2409.06594*, 2024.

[HR24b] Tal Herman and Guy Rothblum. Interactive proofs for general distribution properties. In *2024 IEEE 65th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 528–538. IEEE, 2024.

[HT10] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC '10)*, pages 705–714. ACM, 2010.

[HU14] Moritz Hardt and Jonathan Ullman. Preventing false discovery in interactive data analysis is hard. In *2014 IEEE 55th annual symposium on foundations of computer science*, pages 454–463. IEEE, 2014.

[JLN+19] Christopher Jung, Katrina Ligett, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Moshe Shenfeld. A new analysis of differential privacy's generalization guarantees. *arXiv preprint arXiv:1909.03577*, 2019.

[Kam20] Gautam Kamath. Cs 860: Algorithms for private data analysis, 2020. Lecture notes and videos available at `http://www.gautamkamath.com/CS860-fa2020.html`.

[Kea98] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[KSS94] Michael Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2–3):115–141, 1994.

[MS23] Saachi Mutreja and Jonathan Shafer. Pac verification of statistical algorithms. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 5021–5043. PMLR, 2023.

[MS26] Saachi Mutreja and Jonathan Shafer. Pac verification of statistical algorithms. *Theory of Computing*, 22(5):1–40, 2026. A preliminary version appeared in the Proceedings of COLT'23.

[MT07] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 94–103. IEEE Computer Society, 2007.

[SU15] Thomas Steinke and Jonathan Ullman. Interactive fingerprinting codes and the hardness of preventing false discovery. In *Conference on learning theory*, pages 1588–1628. PMLR, 2015.

[SU16] Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *Journal of Privacy and Confidentiality*, 7(2):3–22, 2016.

[Vad17] Salil Vadhan. The complexity of differential privacy. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.