
AudiFair: Privacy-Preserving Framework for Auditing Fairness

Elisaweta Masserova^{*1} Kshitij Kulkarni^{†-} Antigoni Polychroniadou² Ron D. Rothblum³ Akira Takahashi²

¹Carnegie Mellon University

²J.P. Morgan AlgoCRYPT CoE & AI Research

³Succinct

Abstract

Ensuring fairness in AI is challenging, especially when privacy concerns prevent access to proprietary models and training data. We propose a cryptographic framework for auditing fairness without requiring model disclosure.

Unlike existing solutions – which either fail to capture attack vectors enabling dishonest model providers to pass audits unfairly, or require continuous auditing during deployment involving real-world users – our framework is the first to simultaneously achieve the following:

(1) Model Privacy: Proprietary model details remain hidden from verifiers; **(2) A dishonest model provider cannot cause a non-compliant model to pass the fairness audit without detection by the verifier;** **(3) Pre-deployment Certification:** The certification can be conducted entirely prior to deployment, without requiring interaction with or data from real-world users; **(4) Test Data Transparency:** Test data for auditing is generated in a transparent and accountable way, preventing dishonest parties from manipulating it.

We achieve these goals by carefully combining cryptographic commitments, coin tossing, and zero-knowledge proofs. We evaluate our protocol on decision trees, XGBoost, and neural networks. Our results confirm the practicality of our approach: for a dataset with 10,000 samples and 10 features, and an XGBoost model with 100 trees of depth 3, proof generation takes 42 seconds and verification takes 2 seconds.

1 INTRODUCTION

As AI adoption grows, organizations face increasing pressure to ensure fairness in machine learning systems [Fuster et al., 2022, ProPublica, 2016, Reuters, 2018]. Demonstrating compliance can be difficult, especially when models are proprietary. External examiners often question companies about their training and fairness measures, but this is resource-intensive and yields limited insight. A common alternative is *black-box auditing* [Panigutti et al., 2021, Krafft et al., 2024], where examiners evaluate fairness through API access to the model’s public inference interface. For instance, if a proprietary model performs a classification task, the examiner may query the model with sample input data and analyze the outcomes conditioned on sensitive attributes.

While black-box auditing preserves model confidentiality (up to information revealed by the model’s outputs), it is fundamentally limited: it cannot guarantee that the observed outputs are produced by the claimed model, nor that the same model will be deployed after the audit. This gap creates an opportunity for strategic behavior by the model provider, whose deployment objectives may conflict with the auditor’s goals. For example, the provider may prioritize minimizing default risk by disproportionately rejecting applicants from marginalized groups, even when fairness requirements would mandate equal acceptance rates. To nonetheless pass an audit, a dishonest model provider may manipulate responses during the evaluation—either by selectively modifying outputs or by temporarily substituting a different model—while ultimately deploying a non-compliant model in practice.

These limitations highlight the need for mechanisms that can certify desired properties, e.g., fairness above a given threshold, while keeping model details such as weights private and ensuring that the same model is used both during

^{*}Work conducted during the author’s internship at J.P. Morgan.

[†]Work conducted while the author was affiliated with Succinct.

Table 1: Properties and security guarantees of privacy-preserving fairness certification. ✓ = supported; ✗ = not supported.

Work	Model	Fairness Type	Dishonest Provider Robustness	Test Data Transparency	Pre-deployment Certification
Shamsabadi et al. [2023]	DT/RF	Group	✗	✗	✓
Yadav et al. [2024]	NN	Individual	✓	✗	✗
Segal et al. [2021]	General	Group	✓	✗	✓
Franzese et al. [2024]	General	Group	✓	✗	✗
Bourrée et al. [2025]	General	Group	✗	✗	✓
This work	General	Group	✓	✓	✓

audit and deployment. Recent frameworks for such *privacy-preserving* and *certifiable* ML use cryptographic and differential privacy techniques to address this challenge. Several works target fairness certification in particular, but existing approaches still trade off security, the scope of fairness metrics, or practicality.

Among existing works, only three [Segal et al., 2021, Yadav et al., 2024, Franzese et al., 2024] provide security against *dishonest model providers* (see Table 1) who can deviate from the protocol specification. However, two of the three – OATH Franzese et al. [2024] and Fairproof Yadav et al. [2024] – require client participation in fairness verification, which is problematic since it exposes real users to potentially unfair models and may cause irreversible harm before violations are detected.

Furthermore, none of the three works consider *malicious auditors*, who may deviate from the protocol in order to falsely accuse an honest provider of failing the fairness certification—for example, to damage their reputation or benefit a competitor. In particular, a malicious auditor may engineer an audit dataset that misrepresents the model’s true fairness properties. As a simple example, consider a model whose predictions depend only on the first digit of a ZIP code—a feature that is largely uncorrelated with gender in the real population. Such a model may satisfy fairness criteria on naturally occurring data. However, an adversarial auditor could construct an audit dataset in which the first ZIP code digit is artificially correlated with gender, thereby causing the model to appear unfair despite behaving fairly in practice. This illustrates the need for certification mechanisms that provide robustness not only against dishonest model providers, but also against malicious auditors who may manipulate the audit process itself.

While prior works provide (at least partial) protection against dishonest providers, it does not ensure the integrity of fairness certification in the presence of malicious auditors. More broadly, existing approaches do not simultaneously guarantee robustness against adversarial behavior by both the provider and the auditor while preserving model confidentiality and avoiding reliance on real-world clients.

To address these challenges, we propose AudiFair, a privacy-

preserving framework for certifying model fairness using cryptographic commitments and zero-knowledge proofs. AudiFair prevents potentially dishonest providers from bypassing the fairness checks by requiring them to prove the fairness using a *synthetically generated* fresh test dataset, and does not require involving real-world clients in the auditing process. Along the way, we identify key properties that a privacy-preserving auditing framework should satisfy, and formally prove that AudiFair (to the best of our knowledge, for the first time) achieves all of them:

Model Privacy: Proprietary model parameters and training data remain confidential.

Dishonest Provider Robustness: Even if the model provider deviates from the protocol, the verifier can soundly and verifiably evaluate the model’s fairness using well-established fairness metrics.

Pre-deployment Certification: The certification is established *prior* to deployment; auditing does not require involving model clients.

Test Data Transparency: Neither verifiers nor model providers can arbitrarily deviate from the prescribed test data generation process. This new notion ensures that the test data used for fairness auditing cannot be manipulated to artificially influence the outcome.

Our fairness certification framework is general and can be instantiated with various zero-knowledge proof backends and applied to different model types and fairness metrics. To demonstrate practicality, we implement certification protocols for decision trees, XGBoost Chen and Guestrin [2016], and neural networks, and evaluate them on standard benchmark datasets. Our ZK backend is instantiated with the SP1 Hypercube protocol Succinct Labs [2026], which can efficiently prove the correct execution of any program compiled for the RISC-V architecture. On a dataset with 10,000 data points and 10 features, our protocol for certifying the fairness of XGBoost models with 100 trees and a maximum depth of 3 achieves proof generation in 42 seconds and verification in 2 seconds on an AWS g5.12xlarge instance. Given the widespread use of XGBoost in finance and healthcare, our results demonstrate the practical feasibility of certifying fairness for real-world models.

2 RELATED WORK

The most related to us are OATH and C-PROFITT, which use zero-knowledge proofs to certify (group) fairness. In terms of security, in contrast to us, C-PROFITT does not guarantee fairness if the model provider is dishonest and deviates from the protocol specifications. Specifically, C-PROFITT lets the model provider select the dataset using which fairness is certified, and assumes—rather than enforces—that it matches the training data. Consequently, a dishonest provider can pass fairness checks with an arbitrary “unfair” model by carefully engineering this dataset. While this limitation was recently pointed out in OATH [Franzese et al., 2024], we show an even easier attack in App. C.

To mitigate the risk of dishonest model providers manipulating data, OATH introduces additional parties who supply independent samples. It involves three roles: the client, who provides a data point for classification; the prover (i.e., model provider); and the verifier, who assesses fairness with respect to both the calibration dataset and client queries. The calibration dataset may be selected either by the prover or by the verifier, depending on the implementation. However, when chosen by the prover, the fairness check is again vulnerable to data-forging attacks—same as C-PROFITT. In such cases, OATH falls back to guarantees based solely on the client’s queries. However, another problem is that a malicious client may itself launch a similar attack by choosing their queries in an adversarial way. This is problematic, if, e.g., a malicious client, frustrated by not achieving a desired outcome, seeks to retaliate by framing the prover negatively in the eyes of the verifier. Our test data transparency helps with mitigating this issue.

Further, Segal et al. [2021] proposed a fairness certification framework that, similar to ours, ensures model privacy, robustness against dishonest provers, and provides pre-deployment certification. However, in contrast to us, Segal et al. [2021] does not provide security against adversarial verifiers who attempt to manipulate the audit dataset. Moreover, while their work focuses on image data, our approach applies to general data types—including tabular data—for which suitable synthetic data generation methods exist.

The two works also differ significantly in methodology. Segal et al. [2021] rely on general-purpose secure multi-party computation (MPC), whereas we design a specialized protocol based on cryptographic commitments, secure coin tossing, and zero-knowledge proofs. While Segal et al. [2021] does not provide an implementation, their use of general-purpose MPC would likely incur substantially higher overhead compared to our tailored construction.

Finally, similar to our work, recent studies [Bourrée et al., 2025, Yuan and Wang, 2025] investigate the use of synthetic data for fairness auditing. In contrast to us, however, these works do not provide cryptographic guarantees – in

particular, Bourrée et al. [2025] is vulnerable to model swapping attacks, where the model provider swaps the model during the audit in order to pass the audit, and Yuan and Wang [2025] assumes a fully trusted auditor and lets model provider send the model to this auditor in clear.

We discuss further related works in Appendix B.

3 PRELIMINARIES AND SETTING

Notation We use a bracket notation $[n]$ to denote a set of integers $\{1, \dots, n\}$. We denote a cryptographic hash function by $H(\cdot)$. The number of test data points is n . For any algorithm $\mathcal{A}$, we denote its output on input x by $y \leftarrow \mathcal{A}(x)$. Whenever the randomness ρ is internally sampled by $\mathcal{A}$, we write $y \xleftarrow{\rho} \mathcal{A}(x)$. We denote the process of uniformly sampling an element x from a set S by $x \xleftarrow{\$} S$.

Fairness Over the years, numerous fairness metrics have emerged Heidari et al. [2019], each grounded in distinct philosophical and societal perspectives. These notions often conflict and there is no consensus on a single, “universal” definition of fairness that suits all use cases. We emphasize that it is not our goal to identify the most appropriate notion, nor is it to advertise for any specific definition. Let s denote the sensitive attribute, which is binary in our setting. We focus on the following traditional group-fairness notions:

- **Demographic Parity (DP)** ignores the ground truth and aims to equalize the probability of a positive classifier output in each sensitive group. Let $\hat{Y}$ denote the predicted outcome, then:

$$\text{DP}(\hat{Y}; s) = |\Pr[\hat{Y} = 1 \mid s = 0] - \Pr[\hat{Y} = 1 \mid s = 1]|$$

- **Equalized Odds (EqOd)** aims to equalize the false positive and true positive rates in each sensitive group. Let $\hat{Y}$ denote the predicted outcome, and Y denote the true outcome. Then:

$$\text{EqOd}(\hat{Y}; s) = \max_{y \in \{0,1\}} |\Pr[\hat{Y} = 1 \mid s = 0, Y = y] - \Pr[\hat{Y} = 1 \mid s = 1, Y = y]|$$

We also consider the MRD metric, which is used by major US banks to evaluate the fairness of credit scoring models:

- **Mean Residual Difference (MRD)** measures systematic bias in the model’s prediction errors [Zink and Rose, 2020, Corbett-Davies et al., 2023]. If one group consistently has higher or lower residuals, this suggests unfair treatment.

$$\text{MRD}(\hat{Y}; s) = |E[Y - \hat{Y} \mid s = 0] - E[Y - \hat{Y} \mid s = 1]|$$

Cryptographic Commitment Cryptographic ML certification often uses *commitment schemes* [Blum, 1981]. In

particular, commitments allow a model provider to bind itself to a model in a way that hides model internals, yet prevents the provider from swapping models during or after certification. Formally, a commitment scheme COM is a tuple of the following algorithms:

- A *commitment algorithm* $\text{com} \xleftarrow{\rho} \text{Commit}(\text{msg})$: Takes as input a message msg , internally samples a randomness ρ , and returns a commitment com .
- An *opening algorithm* $b \leftarrow \text{Open}(\text{com}, \text{msg}, \rho)$: Outputs a decision bit $b \in \{0, 1\}$ indicating whether com is a commitment of message msg under randomness ρ .

We require standard security properties: *Hiding* (i.e., com reveals nothing about the committed message m) and *binding* (i.e., the sender cannot open com to two distinct messages m and m'). See Appendix A.3 for a formal treatment.

Non-Interactive Zero-Knowledge Proofs Another key technique in cryptographic ML certification are the *zero-knowledge proofs*. A zero-knowledge proof is a cryptographic protocol between two parties that allows the *prover* to convince the *verifier* that a *statement* is true *without revealing any information beyond its validity*. The statement’s validity is formally defined by an NP relation $\mathcal{R}$: a statement x is valid if there exists a *witness* w such that $(x, w) \in \mathcal{R}$. The statement is public and known to both parties, while the witness is private and only known to the prover. For example, one can prove that a model h outputs y on a public input a *without revealing anything about h ’s weights*. Here, the *statement* is $x = (a, y)$, and the witness is $w = h$.

Formally, a non-interactive zero knowledge proof system (henceforth NIZK) for an NP relation $\mathcal{R}$ is defined as the tuple $\text{ZK} = (\text{Gen}, \text{Prove}, \text{Verify})$:

- $\text{pp} \leftarrow \text{Gen}(1^\lambda)$ is a setup algorithm that, given the security parameter λ^1 , outputs public parameters pp .
- $\pi \leftarrow \text{Prove}(\text{pp}, x, w)$ is a prover algorithm that outputs a proof π asserting $(x, w) \in \mathcal{R}$. If $(x, w) \notin \mathcal{R}$, Prove outputs $\perp$.
- $b \leftarrow \text{Verify}(\text{pp}, x, \pi)$ is a verifier algorithm that outputs a decision bit $b \in \{0, 1\}^*$.

ZK must satisfy standard security properties: *completeness* (i.e., if Prove and Verify follow the protocol, Verify always accepts), *(knowledge) soundness* (i.e., if Verify accepts the proof generated by a cheating prover $\mathcal{A}$, then it must be that $\mathcal{A}$ owns a valid witness w satisfying given NP relation w.r.t. statement x), and *zero knowledge* (i.e., if the proof generated by Prove leaks nothing except that $\exists w$ such that $(x, w) \in \mathcal{R}$). See Appendix A for a formal treatment.

Problem Setting and Threat Model. We consider a setting where a model provider holds a proprietary classification

model that must remain private. An auditor seeks to verify fairness properties of this model—for example, whether its demographic parity gap is below a specified threshold on a uniformly random sample from the target distribution. This verification must be performed prior to the model’s deployment. The model provider may be malicious and attempt to pass the audit with a model that *does not* satisfy the required fairness guarantee. Conversely, the auditor may be malicious and (i) attempt to extract proprietary information about the model or (ii) misrepresent the achieved fairness guarantee, causing the Prover to fail the audit despite holding a fair model. In the following, we will refer to the model provider as *Prover*, and to the auditor as *Verifier*.

4 ABSTRACT FAIRNESS AUDITING FRAMEWORK

We now design a mechanism which allows to prove (prior to deployment) that a proprietary model satisfies certain fairness criteria without revealing anything else about the model. Our framework applies to arbitrary classification models. We first present a high-level overview of our protocol (Section 4.1), provide a formal definition of the auditing framework and its security properties (Section 4.2), and then present a generic construction of our protocol (Section 4.3).

4.1 OVERVIEW

To motivate our design, we first discuss potential pitfalls of naive approaches to certifying fairness in a privacy-preserving manner. Recall that, while the fairness metrics from Section 3 are defined with respect to the underlying data distribution, in practice, one can only estimate them on a finite dataset $\mathcal{D} = \{x_i\}_{i=1}^n$. Thus, the question is how to select this dataset $\mathcal{D}$ for the fairness certification in a way that prevents both the provider and the verifier from manipulating the dataset to their advantage. Having the provider select $\mathcal{D}$ is clearly problematic, as they can simply select an arbitrary dataset on which their model appears to be fair (see also Section C). Another approach is to have the provider and the verifier agree on a well-known public dataset in advance, and then ask the provider for a zero-knowledge proof that M is fair with respect to this dataset. However, this does not address the issue either: a recent work Luck et al. [2026] demonstrated that a dishonest provider can deliberately overfit their model to the agreed-upon dataset, thereby passing the fairness check with a model that behaves unfairly on the actual data distribution.

A more robust approach is to have the verifier generate a fresh dataset $\mathcal{D}$ for every certification process after the provider finalizes the model. However, this presents two challenges. First, in practice, it may be difficult for the auditor to generate fresh data for every audit. More importantly, while allowing the verifier to generate $\mathcal{D}$ prevents the

¹A security parameter determines the desired level of security.

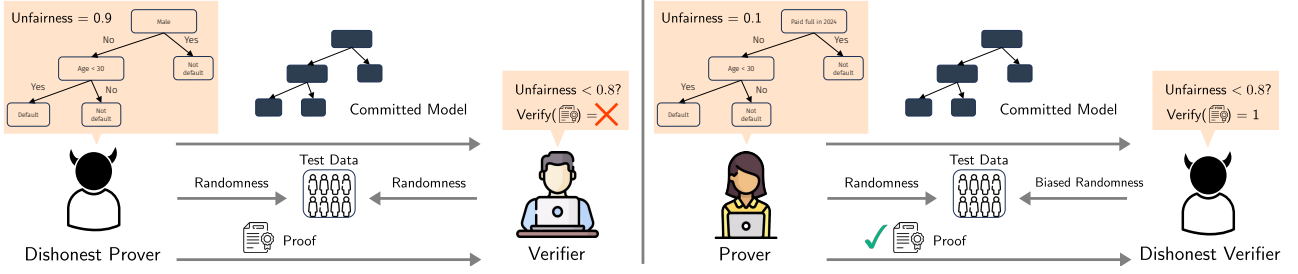


Figure 1: Overview of our framework AudiFair for auditing fairness of proprietary models. **Left:** A dishonest prover with the model that does not pass the fairness check attempts to cheat during the proof generation, but the verifier rejects. **Right:** An honest prover successfully generates a valid fairness proof, even if the verifier is dishonest and attempts to bias the test data. Icons are retrieved from flaticon.com.

provider from overfitting to a fixed dataset, it introduces a new attack vector: a malicious verifier can manipulate the dataset to make a fair model appear unfair, as discussed in the introduction.

To address these challenges, our idea is to require provider to prove the fairness of their model with respect to a *synthetic test dataset jointly sampled by both the provider and the verifier*, followed by a zero-knowledge proof of fairness. Our framework (see also Figure 1) proceeds as follows:

- (a) **Commit:** Model Provider (or Prover) $\mathcal{P}$ generates a commitment com to the model M , and sends com to Model Verifier $\mathcal{V}$.
- (b) **Data Generation:** $\mathcal{P}$ and $\mathcal{V}$ jointly compute and agree on the randomness r for generating the synthetic test dataset $\mathcal{D}$ and the corresponding ground truth labels $\mathcal{Y}$, according to a prescribed and randomized synthetic data generation algorithm DataGen.
- (c) **Prove Fairness:** Using the commitment com , the test dataset $\mathcal{D}$, and the ground truth labels $\mathcal{Y}$, $\mathcal{P}$ and $\mathcal{V}$ conduct zero-knowledge proof that committed M satisfies the fairness condition with respect to $\mathcal{D}$ and $\mathcal{Y}$.

In more detail, in Step (a), the Prover sends a cryptographic commitment com to the model, which binds the prover to the specific model M (while hiding its internals). Crucially, we require that the Prover does so without knowing the exact dataset $\mathcal{D}$ and the corresponding ground truth labels $\mathcal{Y}$ that will be used for the fairness check: If the order is reversed, i.e., a dishonest Prover first receives $\mathcal{D}$ and $\mathcal{Y}$, then it can commit to a model that is overfitted to the exact test dataset, thus cheating the fairness check.

In Step (b), after the Prover and the Verifier jointly sample a uniformly random seed r , they locally derive $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$, where DataGen is a synthetic data generation algorithm that is agreed upon by both parties in advance. In practice, DataGen can be any well-established public synthetic data generation process with hardcoded training data, e.g., CTGAN Xu et al. [2019] consisting of randomized

training of synthesizer followed by randomized sampling process. This step prevents either party from manipulating the data generation process to their advantage². In our concrete instantiation presented later, we realize this step by interleaving Blum’s two-party coin tossing protocol [Blum, 1981]. Alternatively, one could rely on a secure randomness beacon (e.g., of Standards and Technology [2018], Baum et al. [2023]) to obtain fresh, uniform randomness from an external source.

Step (c) can be further decomposed into the following:

Computation of Fairness Score Prover first computes the prediction for the given test dataset $\mathcal{D} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, by evaluating a previously committed model M on each data point $\mathbf{x}_i$. Finally, the prover uses the predicted labels $\hat{\mathcal{Y}} = \{\hat{y}_i\}_{i \in [n]}$ (along with the $\mathcal{D}$ and the corresponding ground-truth $\mathcal{Y}$) to obtain the fairness result. We abstract the process of calculating fairness metric by a function $\text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s)$, where s is a binary sensitive attribute. The function FM can be instantiated in different ways, depending on which fairness metric is employed from Section 3. For instance, to calculate Demographic Parity, one may define

$$\text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) := \left| \frac{|\{\mathbf{x}_i \in \mathcal{D} : \mathbf{x}_i[s] = 0, \hat{y}_i = 1\}|}{|\{\mathbf{x}_i \in \mathcal{D} : \mathbf{x}_i[s] = 0\}|} - \frac{|\{\mathbf{x}_i \in \mathcal{D} : \mathbf{x}_i[s] = 1, \hat{y}_i = 1\}|}{|\{\mathbf{x}_i \in \mathcal{D} : \mathbf{x}_i[s] = 1\}|} \right| \quad (1)$$

Prover then checks that $\text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) \leq t$ with respect to a pre-agreed threshold t between the Prover and the Verifier.

Zero-knowledge Fairness Proof Assuming that the fairness check passes, in the last step, the Prover uses a secure NIZK scheme on the predicted labels $\hat{\mathcal{Y}}$ and fairness score v , along with the data $\mathcal{D}$, ground-truth $\mathcal{Y}$, and commitment com to

²see Section 5.3 and Section E.6 for corresponding experiments and discussion

show the following statement in zero-knowledge: I know a model M for which all of the following holds:

- (1) It is committed under com
- (2) M 's prediction for $\mathbf{x}_i$ corresponds to $\hat{y}_i$ for each $i \in [n]$
- (3) When the agreed upon fairness metric function FM is computed on the tuple $(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}} = \{\hat{y}_i\}_{i \in [n]}, s)$, the outcome v is below threshold t .

Note that our protocol allows for a flexible choice of the ZK scheme to prove these conditions. Further, note that because we use a secure ZK, the protocol does not leak predictions $\hat{\mathcal{Y}}$ w.r.t test data $\mathcal{D}$; Prover only leaks the fact that a committed model M meets the fairness condition w.r.t. a given test data set, sensitive attribute, and fairness threshold.

Extension 1: Proving exact fairness scores Above, we focused on the case where the Prover hides the exact outcome of FM and the Verifier only learns a binary outcome (i.e., whether the committed model is fair or not with respect to a public threshold t). If the Verifier is interested in understanding the strengths or failings of the committed model, then one can easily tweak our protocol by having the Prover reveal a fairness score v and generate a proof that $v = \text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s)$ holds in Step (3). As this merely skips a comparison operation, the performance of this variant is the same as the original protocol.

Extension 2: Supporting non-binary sensitive attributes While we focus on binary sensitive attributes for simplicity, it is possible to extend our protocol to non-binary attributes. For this, we compute the fairness score by simply comparing *each* protected minority group to the majority group, and taking the maximum of the (absolute) differences.

4.2 DEFINITION OF PRIVACY-PRESERVING FAIRNESS AUDITING

We formally define the syntax and the security properties of the auditing framework sketched above. While zero-knowledge proofs alone (Section 3) do not specify *how* and *when* a public statement $\mathbf{x}$ is generated, our tailored syntax and security notions precisely define the requirements for generating $\mathbf{x}$ —consisting of a commitment to the model and the synthetic data—in a specific way to prevent manipulation of the fairness metric.

Definition 1. Define an NP relation for fairness auditing parameterized by a sensitive attribute s , fairness threshold t , and fairness metric FM:

$$\mathcal{R}_{\text{fair}} = \left\{ ((\mathcal{D}, \mathcal{Y}), M) : \begin{array}{l} \hat{\mathcal{Y}} = M(\mathcal{D}) \\ \wedge \text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) \leq t \end{array} \right\}$$

Let DataGen be a randomized synthetic data generation algorithm. Let COM = (Commit, Open) be a commitment

scheme that supports committing to a model M using randomness ρ . Let Setup(1^λ) be a setup algorithm that generates public parameters pp.

A privacy-preserving fairness auditing protocol Π for $\mathcal{R}_{\text{fair}}$ and DataGen consists of Setup, COM, and the following interactive Audit process. In Audit, the model provider $\mathcal{P}$ receives (M, ρ) as private input, while both $\mathcal{P}$ and the verifier $\mathcal{V}$ receive the public parameters pp and commitment com as public inputs. $\mathcal{P}$ and $\mathcal{V}$ interact in a way that $\mathcal{P}$ eventually halts by locally outputting a pair $(\mathcal{D}_{\mathcal{P}}, \mathcal{Y}_{\mathcal{P}})$ of synthetic data and ground truth labels, while $\mathcal{V}$ halts by locally outputting synthetic data and ground truth $(\mathcal{D}_{\mathcal{V}}, \mathcal{Y}_{\mathcal{V}})$, and a decision bit $b \in \{0, 1\}$.

We state the security properties of Π informally below and defer the formal definitions to Appendix F.

Completeness requires that if both the Prover and Verifier honestly follow the protocol, and the Prover owns a fair model M , then the Verifier is always convinced *and* they agree on the same synthetic data.

Binding: Once the Prover commits to a model, it cannot change it without detection.

Dishonest Provider Robustness: The protocol is robust against dishonest model providers, meaning that if $\mathcal{P}$ tries to cheat by committing to an unfair model M (i.e., not satisfying the fairness condition described by $\mathcal{R}_{\text{fair}}$), the $\mathcal{V}$ can detect it by outputting $b = 0$.

Model Privacy: The protocol leaks no information about the model M to $\mathcal{V}$ except that it satisfies the fairness condition described by $\mathcal{R}_{\text{fair}}$.

Test Data Transparency: Neither party can bias the distribution of the test dataset $\mathcal{D}$ generated during the interactive protocol, provided the protocol does not abort.

Note that any protocol satisfying the above definition inherently provides **pre-deployment certification**: the interaction between $\mathcal{P}$ and $\mathcal{V}$ can be carried out prior to deployment and does not require involving real-world users.

Assumption on the Synthetic Data Generator Similar to Segal et al. [2021], our framework relies on the assumption that DataGen produces synthetic data that faithfully represents the real data distribution X with respect to the fairness metric of interest. Intuitively, this captures the requirement that a model's fairness outcome on synthetic data should match its outcome on real data drawn from X , with high probability. Note that this assumption is weaker than the standard notion of computational indistinguishability in cryptography, as it only requires the fairness outcomes to be indistinguishable with respect to a limited class of models, rather than requiring that no adversary can distinguish the entire distributions of synthetic and real data. We defer the formal definition of this assumption to Appendix F.

Protocol 1: Generic Construction of Π

Setup(1^λ): Run $\text{pp} \leftarrow \text{ZK.Gen}(1^\lambda)$. Output pp .

Interactive Audit process: $\mathcal{P}$ and $\mathcal{V}$ run the following protocol sequentially. If any check fails, the protocol aborts and $\mathcal{V}$ halts by outputting $b = 0$.

- 1: $\mathcal{V}$ sends $h_2 = H(r_2)$, where $r_2 \leftarrow \{0, 1\}^k$
- 2: $\mathcal{P}$ sends $h_1 = H(r_1)$, where $r_1 \leftarrow \{0, 1\}^k$
- 3: $\mathcal{V}$ sends r_2
- 4: $\mathcal{P}$ checks that $h_2 = H(r_2)$ and does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_\mathcal{P}, \mathcal{Y}_\mathcal{P}) \leftarrow \text{DataGen}(r)$
 - (c) check $\text{FM}(\mathcal{D}_\mathcal{P}, \mathcal{Y}_\mathcal{P}, M(\mathcal{D}_\mathcal{P}), s) \leq t$
 - (d) $\pi \leftarrow \text{ZK.Prove}(\text{pp}, (\text{com}, \mathcal{D}_\mathcal{P}, \mathcal{Y}_\mathcal{P}), (M, \rho))$
 - (e) send (π, r_1) to $\mathcal{V}$
 - (f) locally output $(\mathcal{D}_\mathcal{P}, \mathcal{Y}_\mathcal{P})$
- 5: $\mathcal{V}$ checks that $h_1 = H(r_1)$ and does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}) \leftarrow \text{DataGen}(r)$
 - (c) run $b \leftarrow \text{ZK.Verify}(\text{pp}, (\text{com}, \mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}), \pi)$
 - (d) locally output $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}, b)$

4.3 GENERIC CONSTRUCTION AND SECURITY

We provide a generic construction of privacy-preserving fairness auditing protocol in Protocol 1. Other than COM, the protocol utilizes the following cryptographic building blocks as subroutines: hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\ell$, and non-interactive zero-knowledge (NIZK) proof system $\text{ZK} = (\text{Gen}, \text{Prove}, \text{Verify})$ for “commit-and-prove” relation defined as

$$\mathcal{R}_{\text{cp}} := \left\{ ((\text{com}, \mathcal{D}, \mathcal{Y}), (M, \rho)) : \begin{array}{l} \text{Open}(\text{com}, M, \rho) = 1 \\ \wedge ((\mathcal{D}, \mathcal{Y}), M) \in \mathcal{R}_{\text{fair}} \end{array} \right\}$$

We now state the main security claim of our generic construction in Theorem 1. The formal statement and proof is deferred to Appendix F. Intuitively, dishonest provider robustness is realized by the fact that a dataset is obtained after the Prover commits to a model, and that knowledge soundness of ZK forces the Prover to use a model that was committed earlier and satisfies the fairness condition. Model privacy is achieved by secure instantiation of COM and ZK. Finally, data transparency is achieved by the exchange of hashes of randomness shares at the beginning of the protocol, which ensures that neither party can bias the randomness used to generate the test dataset.

Theorem 1 (informal). *Suppose ZK and COM satisfy the standard properties defined in App. A.3 and A.4, and the hash function H is modeled as a programmable random oracle (App. A.2). Then, Protocol 1 satisfies completeness, binding, dishonest provider robustness, model privacy, and*

test data transparency.

5 CONCRETE INSTANTIATIONS AND EVALUATIONS

5.1 OVERVIEW OF INSTANTIATIONS

We now describe how to instantiate our protocol for certifying decision trees, XGBoost, and feed-forward neural networks (FFNN). Each building block of Protocol 1 from the previous section is instantiated as follows:

ZK Proof We utilize SP1 Succinct Labs [2026] Hypercube as a backend for our NIZK. This proof system is based on an interactive oracle proof (IOP) Ben-Sasson et al. [2016], Reingold et al. [2021], which is then compiled into a succinct non-interactive zero-knowledge argument. The proof system is instantiated for the Rust script checking $\mathcal{R}_{\text{cp}}((\mathcal{D}, \mathcal{Y}), (M, \rho)) = 1$ compiled for the RISC-V architecture. Our implementation validates all three fairness metrics considered in Section 3 within a single proof. Background on SP1 and further implementation details are provided in Appendix D.

Hash Function We use SHA2-256 for the hash function H in coin tossing and for a commitment COM to private model parameters. To ensure hiding of COM, we use a random 256-bit nonce before hashing the model parameters.

Data Generation We use CTGAN [Xu et al., 2019] from the SDV library [Developers, 2024] as a synthetic generator.

5.2 BENCHMARKING ZK PROOF OF FAIRNESS

We benchmarked the performance of ZK for checking $\mathcal{R}_{\text{cp}}$ on AWS EC2 g5.12xlarge instances on AWS EC2 g5.12xlarge instances with 4 GPUs and 96GB GPU memory. We considered the following models: (1) a single decision tree with depth $h = 10$, (2) XGBoost models with $(h, K) = (50, 5)$ and $(100, 3)$, where K denotes the number of trees, and (3) a feed-forward neural network (FFNN) with two hidden layers of size 64 and 32, respectively. All the models were trained on the 50% of the standard datasets: Default Credit, Adult, and ACS Income (see Section E.1). Table 2 summarizes the performance of our ZK instantiation for different models trained on ACSIncome. In Appendix E, we present the performance for models trained on the Default Credit and Adult datasets, as well as additional benchmarks for XGBoost on larger number of samples and varying parameters h and K . The appendix further reports experiments on the scalability of our approach to larger FFNNs with $\sim 10,000$ parameters (Section E.3), and a performance comparison with the baseline schemes OATH Franzese et al. [2024] and C-PROFIT Shamsabadi et al. [2023] (Section E.4), showing that our scheme achieves substantially faster verification than both baselines. Our results show that

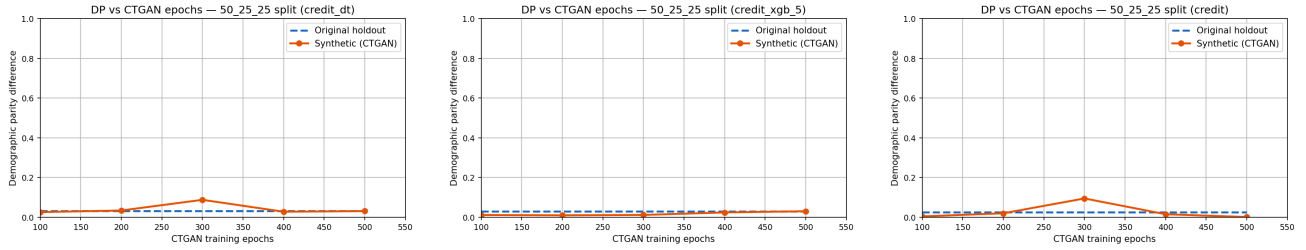


Figure 2: Comparison of the Demographic Parity score Δ computed on (a portion of) DefaultCredit dataset and on CTGAN-generated synthetic data as a function of the number of CTGAN training epochs, with gender as the sensitive attribute. The models considered are: a decision tree (left), XGBoost (center), and a feed-forward neural network (right). The baseline Demographic Parity score Δ (blue dashed line) is evaluated on the reference holdout set. CTGAN is trained on its designated portion to generate synthetic datasets of the same size as the reference holdout set; these are then used to evaluate Δ for the same fixed classifier M (orange solid line).

proving fairness of decision trees and XGBoost ensembles with practically relevant parameters can be efficient, while auditing FFNNs is more expensive but still within the realm of practicality.

5.3 EVALUATING FAIRNESS VIA SYNTHETIC DATA

We conduct experiments to assess whether synthetic data can serve as a practical substitute for original test data when evaluating fairness. The experimental setup is as follows. The dataset is partitioned into three disjoint portions: 50% for training the classifier, 25% as a reference holdout set for computing a baseline fairness score, and 25% for training the CTGAN synthesizer (see Section E.6 for additional experiments that use TVAE and Gaussian Copula). The classifier M is trained on the training portion and evaluated on the reference holdout to obtain the baseline fairness score. We then train CTGAN on its designated portion, generate synthetic data, and evaluate the same fixed classifier M on the synthetic dataset.

We evaluate both demographic parity and equalized odds across multiple model classes: a decision tree of depth 10, XGBoost with 50 trees of depth 5, and a feed-forward neural network with two hidden layers of sizes 64 and 32. We show the results for demographic parity on the Default Credit dataset in Figure 2, and defer the remaining plots to Section E.6. Overall, we find that synthetic data can serve as a reliable proxy: for most dataset-model combinations, once CTGAN has been trained for sufficiently many (~ 400 -450) epochs, the difference between the baseline and synthetic fairness scores is below 5%, and it never exceeds 13%.

In terms of runtime, CTGAN training and subsequent synthetic data generation took 172 seconds for ACS Income (restricted to the first 40,000 rows), 198 seconds for Default Credit, and 233 seconds for Adult.

6 CONCLUSION, LIMITATIONS, AND FUTURE WORK

We proposed a secure solution to prove that a (proprietary and private) decision tree model satisfies given fairness constraints. Our solution is the first to simultaneously achieve model privacy, dishonest provider robustness, pre-deployment certification, and test data transparency.

In terms of limitations, we note that each audit reveals whether the committed model satisfies the specified fairness threshold. This is per design: learning this outcome is precisely the purpose of the audit. However, repeated adaptive audits leak information through multiple such pass/fail outcomes. This is a generic concern for any repeated audit interface. In deployment, the audit predicate, threshold, and the number of audits should therefore be fixed in advance. Furthermore, we note that AudiFair certifies a specific model snapshot. Supporting post-audit model updates would require lightweight re-certification mechanisms, which are a natural and important extension for production ML pipelines but orthogonal to the present work.

We leave several interesting questions for future work: In this work, we assume that a model’s fairness outcome on synthetic data output by DataGen closely matches its outcome on real data. While our empirical evaluation suggests that synthetic data can serve as a proxy for the original data when assessing fairness of certain models, an interesting direction for future work is to study how well the induced synthetic distribution approximates the true data distribution, what class of models this approximation holds for, and whether the usage of synthetic data instead of original one can be feasible to audit properties such as, e.g., accuracy. Further, it would be interesting to provide a solution in which the verifier’s test data remains *private* while protecting against dishonest behaviors. General-purpose solutions for this scenario such as secure multi-party computation exist, but these are not sufficiently efficient. Obtaining a practical scheme likely requires developing new cryptographic

Table 2: Performance of our ZK instantiation for checking Demographic Parity/Equalized Odds/Mean Residual Difference on synthetic samples generated based on the ACSIncome dataset. h is the height of each tree and K is the number of trees. The experiment was conducted on an Amazon EC2 g5.12xlarge instance with 4 GPUs and 96GB GPU memory.

Model	Params	Samples	Prove (s)	Verify (s)	Proof Size (MB)
Decision Tree	$h = 10$	5,000	8	0.3	7.1
		10,000	13	0.5	11.4
XGBoost	$h = 3, K = 100$	5,000	23	1.0	21.5
		10,000	42	1.9	39.0
XGBoost	$h = 5, K = 50$	5,000	18	0.8	17.2
		10,000	32	1.4	30.2
Neural Net	2,784	5,000	410	21.8	453.5
		10,000	872	59.5	1235.2

techniques.

ACKNOWLEDGMENT & DISCLAIMER

This paper was prepared in part for information purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co and its affiliates (“JP Morgan”), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

- Kasra Abbaszadeh, Christodoulos Pappas, Jonathan Katz, and Dimitrios Papadopoulos. Zero-knowledge proofs of training for deep neural networks. In *CCS*, pages 4316–4330. ACM, 2024.
- Carsten Baum, Bernardo David, Rafael Dowsley, Ravi Kishore, Jesper Buus Nielsen, and Sabine Oechsner. CRAFT: Composable randomness beacons and output-independent abort MPC from time. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 439–470. Springer, Heidelberg, May 2023. doi: 10.1007/978-3-031-31368-4_16.
- Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO’93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994. doi: 10.1007/3-540-48329-2_21.
- Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016. doi: 10.1007/978-3-662-53644-5_2.
- Manuel Blum. Coin flipping by telephone. In Allen Gersho, editor, *CRYPTO’81*, volume ECE Report 82-04, pages 11–15. U.C. Santa Barbara, Dept. of Elec. and Computer Eng., 1981.
- Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988. doi: 10.1145/62212.62222.
- Jade Garcia Bourrée, Hadrien Lautreite, Sébastien Gambs, Gilles Trédan, Erwan Le Merrer, and Benoît Rottembourg. P2NIA: privacy-preserving non-iterative auditing. *CoRR*, abs/2504.00874, 2025.
- Jade Garcia Bourrée, Hadrien Lautreite, Sébastien Gambs, Gilles Tredan, Erwan Le Merrer, and Benoît Rottembourg. P2nia: Privacy-preserving non-iterative auditing, 2025. URL <https://arxiv.org/abs/2504.00874>.
- Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM, 2016. doi: 10.1145/2939672.2939785. URL <https://doi.org/10.1145/2939672.2939785>.
- Sam Corbett-Davies, Johann D. Gaebler, Hamed Nilforoshan, Ravi Shroff, and Sharad Goel. The measure and mismeasure of fairness. *J. Mach. Learn. Res.*, 24: 312:1–312:117, 2023. URL <http://jmlr.org/papers/v24/22-1511.html>.

- SDV Developers. Sdv: Synthetic data vault library. <https://github.com/sdv-dev/SDV>, 2024. Version 1.12.0.
- Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. In Marc’Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6478–6490, 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/32e54441e6382a7fbacbbbf3c450059-Abstract.html>.
- Olive Franzese, Ali Shahin Shamsabadi, and Hamed Haddadi. OATH: efficient and flexible zero-knowledge proofs of end-to-end ML fairness. *CoRR*, abs/2410.02777, 2024. doi: 10.48550/ARXIV.2410.02777. URL <https://doi.org/10.48550/arXiv.2410.02777>.
- Andreas Fuster, Paul Goldsmith-Pinkham, Tarun Ramadorai, and Ansgar Walther. Predictably unequal? the effects of machine learning on credit markets. *The Journal of Finance*, 77(1):5–47, 2022.
- Sanjam Garg, Aarushi Goel, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, Guru-Vamsi Policharla, and Mingyuan Wang. Experimenting with zero-knowledge proofs of training. In *CCS*, pages 1880–1894. ACM, 2023.
- Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. ISBN 0-521-79172-3 (hardback).
- Hoda Heidari, Michele Loi, Krishna P. Gummadi, and Andreas Krause. A moral framework for understanding fair ML through economic models of equality of opportunity. In *Conference on Fairness, Accountability, and Transparency*, 2019. URL <https://doi.org/10.1145/3287560.3287584>.
- Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, third edition, 2020. ISBN 9780367331586.
- Tobias D. Krafft, Marc P. Hauer, and Katharina Anna Zweig. Black-box testing and auditing of bias in ADM systems. *Minds Mach.*, 34(2):15, 2024. doi: 10.1007/S11023-024-09666-0. URL <https://doi.org/10.1007/s11023-024-09666-0>.
- Yehuda Lindell. How to simulate it – a tutorial on the simulation proof technique. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography*, volume 5559 of *Lecture Notes in Computer Science*, pages 277–346. Springer, 2009. doi: 10.1007/978-3-642-04159-4_8. URL https://doi.org/10.1007/978-3-642-04159-4_8.
- Carter Luck, Olive Franzese, Elisaweta Masserova, Akira Takahashi, Antigoni Polychroniadou, and Nicolas Papernot. Certified in theory, broken in practice: Assumption gaps in cryptographic model certification. In *USENIX Security 2026*. USENIX Association, 2026.
- National Institute of Standards and Technology. The nist randomness beacon 2.0, 2018. URL <https://csrc.nist.gov/Projects/interoperable-randomness-beacons/beacon-20>.
- Cecilia Panigutti, Alan Perotti, André Panisson, Paolo Bajardi, and Dino Pedreschi. Fairlens: Auditing black-box clinical decision support systems. *Inf. Process. Manag.*, 58(5):102657, 2021. doi: 10.1016/J.IPM.2021.102657. URL <https://doi.org/10.1016/j.ipm.2021.102657>.
- Christodoulos Pappas and Dimitrios Papadopoulos. Sparrow: Space-efficient zkSNARK for data-parallel circuits and applications to zero-knowledge decision trees. In *CCS*, pages 3110–3124. ACM, 2024.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python, 2011. URL <https://scikit-learn.org/>.
- ProPublica. Machine bias: There’s software used across the country to predict future criminals. and it’s biased against blacks., 2016. URL <https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing>.
- Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum. Constant-round interactive proofs for delegating computation. *SIAM J. Comput.*, 50(3), 2021. doi: 10.1137/16M1096773. URL <https://doi.org/10.1137/16M1096773>.
- UCI Machine Learning Repository. Census income, 1996. URL <https://doi.org/10.24432/C5GP7S>.
- UCI Machine Learning Repository. Default of credit card clients, 2009. URL <https://doi.org/10.24432/C5GP7S>.
- UCI Machine Learning Repository. Bank marketing, 2014. URL <https://doi.org/10.24432/C5K306>.
- Reuters. Insight - amazon scraps secret ai recruiting tool that showed bias against women, 2018. URL <https://www.reuters.com/article/world/insi>

ght-amazon-scrapers-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK0AG/.

- Shahar Segal, Yossi Adi, Benny Pinkas, Carsten Baum, Chaya Ganesh, and Joseph Keshet. Fairness in the eyes of the data: Certifying machine-learning models. In Marion Fourcade, Benjamin Kuipers, Seth Lazar, and Deirdre K. Mulligan, editors, *AIES '21: AAAI/ACM Conference on AI, Ethics, and Society, Virtual Event, USA, May 19-21, 2021*, pages 926–935. ACM, 2021. doi: 10.1145/3461702.3462554. URL <https://doi.org/10.1145/3461702.3462554>.
- Ali Shahin Shamsabadi, Sierra Calanda Wyllie, Nicholas Franzese, Natalie Dullerud, Sébastien Gambis, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential-profit: Confidential proof of fair training of trees. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL <https://openreview.net/pdf?id=iIfDQVyuFD>.
- Succinct Labs. SP1 library. <https://docs.succinct.xyz/docs/sp1/introduction>, 2026.
- Haochen Sun, Tonghe Bai, Jason Li, and Hongyang Zhang. zkdl: Efficient zero-knowledge proofs of deep learning training. *IEEE Trans. Inf. Forensics Secur.*, 20:914–927, 2025.
- Suppakit Waiwitlikhit, Ion Stoica, Yi Sun, Tatsunori Hashimoto, and Daniel Kang. Trustless audits without revealing data or models. In *ICML*. OpenReview.net, 2024.
- Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In *Advances in Neural Information Processing Systems*, 2019.
- Chhavi Yadav, Amrita Roy Chowdhury, Dan Boneh, and Kamalika Chaudhuri. Fairproof : Confidential and certifiable fairness for neural networks. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024. URL <https://openreview.net/forum?id=EKye56rLuv>.
- Chhavi Yadav, Evan Laufer, Dan Boneh, and Kamalika Chaudhuri. Expproof : Operationalizing explanations for confidential models with zkps. *CoRR*, abs/2502.03773, 2025.
- Chih-Cheng Rex Yuan and Bow-Yaw Wang. Quantitative auditing of AI fairness with differentially private synthetic data. *CoRR*, abs/2504.21634, 2025. doi: 10.48550/ARXIV.2504.21634. URL <https://doi.org/10.48550/arXiv.2504.21634>.

Jiaheng Zhang, Zhiyong Fang, Yupeng Zhang, and Dawn Song. Zero knowledge proofs for decision tree predictions and accuracy. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 2039–2053. ACM Press, November 2020. doi: 10.1145/3372297.3417278.

Tianyu Zhang, Shen Dong, Oyku Deniz Kose, Yanning Shen, and Yupeng Zhang. Fairzk: A scalable system to prove machine learning fairness in zero-knowledge. In *SP*, pages 3460–3478. IEEE, 2025.

Anna Zink and Sherri Rose. Fair regression for health care spending. *Biometrics*, 76(3):973–982, January 2020. ISSN 1541-0420. doi: 10.1111/biom.13206. URL <http://dx.doi.org/10.1111/biom.13206>.

A EXTENDED PRELIMINARIES

A.1 BASIC NOTATION AND DEFINITIONS IN CRYPTOGRAPHY

In this section, we include minimal introductory material on the basic notation and definitions used in the paper. For a more comprehensive introduction to cryptography, we refer the reader to standard textbooks such as Katz and Lindell [2020], Goldreich [2001].

For two bit strings of equal length, we denote $a \oplus b$ as the bitwise XOR of a and b . For any algorithm $\mathcal{A}$, we denote its output on input x by $y \leftarrow \mathcal{A}(x)$. Whenever the randomness ρ is sampled by $\mathcal{A}$ internally, we write $y \xleftarrow{\rho} \mathcal{A}(x)$. We denote the process of uniformly sampling an element x from a set S by $x \xleftarrow{\$} S$. Let $\lambda \in \mathbb{N}$ be a security parameter. We denote the function $f(\lambda)$ is *negligible* if for every polynomial $p(\lambda)$, there exists λ_0 such that for all $\lambda \geq \lambda_0$, $f(\lambda) < \frac{1}{p(\lambda)}$. A function $f(\lambda)$ is said to be *overwhelming* if $1 - f(\lambda)$ is negligible.

By PPT, we mean *probabilistic polynomial time*. For any probability distributions $\mathcal{D}_0(\lambda)$ and $\mathcal{D}_1(\lambda)$, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally *indistinguishable* if for any PPT algorithm $\mathcal{A}$, the following is negligible in λ :

$$|\Pr [1 \leftarrow \mathcal{A}(1^\lambda, \mathcal{D}_0(\lambda))] - \Pr [1 \leftarrow \mathcal{A}(1^\lambda, \mathcal{D}_1(\lambda))]|$$

If the above holds for *any* algorithm $\mathcal{A}$ (including unbounded ones), we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are statistically indistinguishable. If the above quantity is 0, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are perfectly indistinguishable.

A.2 RANDOM ORACLE MODEL

In this work, we assume that the hash function H is modeled as a *random oracle* Bellare and Rogaway [1994]. In the random oracle model, H is a function that maps arbitrary-length inputs to uniformly random outputs of a fixed length. In this model, H is assumed to be a public function that is accessible to all parties, including the adversary. Thus, whenever a proof of security is in the random oracle model, it is assumed that a reduction or a simulator can observe queries made by an adversary to the random oracle. The random oracle model is an idealized abstraction of hash functions commonly used in security proofs, allowing for the design of secure and efficient protocols. Importantly, whenever H receives a new query, it responds with a freshly and uniformly sampled value $h \xleftarrow{\$} \{0, 1\}^{\ell(\lambda)}$. Moreover, the random oracle is often assumed to be *programmable*, meaning that a reduction or a simulator can predefine the output of the random oracle for a specific input, as long as the input has not been queried before.

A.3 SECURITY PROPERTIES OF CRYPTOGRAPHIC COMMITMENTS

Our protocol relies on a cryptographic *commitment scheme* Blum [1981]. A commitment scheme COM is a tuple of the following algorithms:

- A *commitment algorithm* $\text{com} \xleftarrow{\rho} \text{Commit}(\text{msg})$: Takes as input a message $\text{msg} \in \{0, 1\}^{\ell_m}$ from a message space $\mathcal{M}$, internally samples a randomness $\rho \in \{0, 1\}^{\ell_r(\lambda)}$, and returns a commitment $\text{com} \in \{0, 1\}^{\ell_c(\lambda)}$. Here ℓ_m, ℓ_r, ℓ_c are some polynomials in λ , the security parameter.
- An *opening algorithm* $b \leftarrow \text{Open}(\text{com}, \text{msg}, \rho)$: Outputs a decision bit $b \in \{0, 1\}$ indicating whether an opening of the commitment is valid or not.

We recall standard security properties of cryptographic commitment schemes.

Completeness COM is *complete* if for any message msg in the message space,

$$\Pr [1 \leftarrow \text{Open}(\text{com}, \text{msg}, \rho) : \text{com} \xleftarrow{\rho} \text{Commit}(\text{msg})] = 1$$

Binding COM is *binding* if for any PPT adversary $\mathcal{A}$, the following probability is negligible in λ :

$$\Pr \left[\begin{array}{l} M \neq M' \\ \wedge b = 1 \\ \wedge ab' = 1 \end{array} : \begin{array}{l} (\text{com}, M, \rho, M', \rho') \leftarrow \mathcal{A}(1^\lambda) \\ b \leftarrow \text{Open}(\text{com}, M, \rho) \\ b' \leftarrow \text{Open}(\text{com}, M', \rho') \end{array} \right].$$

Hiding COM is *hiding* if for any PPT adversary $\mathcal{A}$ and any pair of messages $(\text{msg}, \text{msg}')$ in the message space, the following is negligible in λ :

$$\varepsilon_{\text{hide}} = \left| \begin{array}{l} \Pr [1 \leftarrow \mathcal{A}(\text{com}) : \text{com} \leftarrow \text{Commit}(\text{msg})] \\ - \Pr [1 \leftarrow \mathcal{A}(\text{com}) : \text{com} \leftarrow \text{Commit}(\text{msg}')] \end{array} \right|$$

A.4 SECURITY PROPERTIES OF ZERO KNOWLEDGE PROOFS

A (non-interactive) zero knowledge proof system (henceforth NIZK) is defined for *NP relation* $\mathcal{R}$, i.e., a set of *public statements* x and *private witnesses* w , where the size of w is bounded by a polynomial in the size of x , and $(x, w) \in \mathcal{R}$ can be checked in polynomial time given (x, w) . It is denoted by a tuple $\text{ZK} = (\text{Gen}, \text{Prove}, \text{Verify})$ of three algorithms:

- $\text{pp} \leftarrow \text{Gen}(1^\lambda)$ is a setup algorithm that samples a public parameter pp , where λ denotes a security parameter.
- $\pi \leftarrow \text{Prove}(\text{pp}, x, w)$ is a prover that outputs a proof π asserting $(x, w) \in \mathcal{R}$. If $(x, w) \notin \mathcal{R}$, Prove outputs $\perp$.
- $b \leftarrow \text{Verify}(\text{pp}, x, \pi)$ is a verifier that outputs a decision bit $b \in \{0, 1\}^*$.

The common input x to both parties is called *statement*, and Prove 's private input w is called *witness*. For instance, if the prover is tasked to prove $\mathcal{T}(a) = y$ for public a and y for a private decision tree $\mathcal{T}$, one can set $x = (a, y)$ and $w = \mathcal{T}$.

We recall standard security properties of (non-interactive) zero knowledge proof systems.

Completeness ZK is *complete* if for any $(x, w) \in \mathcal{R}$,

$$\Pr \left[1 \leftarrow \text{Verify}(\text{pp}, x, \pi) : \begin{array}{l} \text{pp} \leftarrow \text{Gen}(1^\lambda); \\ \pi \leftarrow \text{Prove}(\text{pp}, x, w) \end{array} \right] = 1$$

Knowledge Soundness ZK is (adaptively) *knowledge sound* if for any PPT adversary $\mathcal{A}$, there exists a (non-blackbox) polynomial time extractor $\text{Ext}_{\mathcal{A}}$ such that the following probability negligible in λ :

$$\Pr \left[b = 1 \wedge (x, w) \notin \mathcal{R} : \begin{array}{l} \text{pp} \leftarrow \text{Gen}(1^\lambda); (x, \pi) \leftarrow \mathcal{A}(\text{pp}) \\ b \leftarrow \text{Verify}(\text{pp}, x, \pi); w \leftarrow \text{Ext}_{\mathcal{A}}(\text{pp}) \end{array} \right]$$

Zero Knowledge ZK is *zero knowledge* if there exists a PPT simulator $\text{Sim} = (\text{Sim}_1, \text{Sim}_2)$ such that for any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ where $\mathcal{A}_1$ always outputs (x, w) satisfying $(x, w) \in \mathcal{R}$, the following is negligible in λ :

$$\varepsilon_{\text{zk}} = \left| \Pr \left[b = 1 : \begin{array}{l} \text{pp} \leftarrow \text{Gen}(1^\lambda) \\ (x, w, \text{st}) \leftarrow \mathcal{A}_1(\text{pp}) \\ \pi \leftarrow \text{Prove}(\text{pp}, x, w) \\ b \leftarrow \mathcal{A}_2(\text{st}, \pi) \end{array} \right] - \Pr \left[b = 1 : \begin{array}{l} (\text{pp}, \text{td}) \leftarrow \text{Sim}_1(1^\lambda) \\ (x, w, \text{st}) \leftarrow \mathcal{A}_1(\text{pp}) \\ \pi \leftarrow \text{Sim}_2(\text{td}, x) \\ b \leftarrow \mathcal{A}_2(\text{st}, \pi) \end{array} \right] \right|.$$

Note that the simulator Sim does not have access to the witness w ; instead, it has the ability to sample a *trapdoor* td that allows it to generate a proof π for any true statement x .

A.5 INTUITION FOR ZERO KNOWLEDGE PROOFS

In this part, we provide an informal overview of zero knowledge proofs. Consider two parties – a prover and a verifier. The verifier is a color blind person. The prover possesses two objects that are identical in every way except, potentially, their color. The prover claims that the colors of these objects are *different*, but does not want to let the verifier know the colors. How can a color blind verifier check whether the prover’s statement is indeed true, i.e., the two otherwise identical objects differ in color? Consider the following protocol:

- First, the prover places both items next to each other on a table.
- Then, the prover leaves the room. The verifier can now flip a coin to decide whether to swap the two objects or not.
- The prover returns and has to declare whether the objects were swapped or not.

At this point, there are two possible scenarios: Either the prover’s claim is true and the objects have different colors, or the claim is false and the objects are identical. In the first case, the prover can memorize the color of the left object when leaving the room, and once they return the prover simply checks whether the color of the left object is still the same as before or not. This way, the prover can always pass the check. If, however, the prover was lying and the two objects are completely identical, they have only a $\frac{1}{2}$ chance of *guessing* whether the verifier swapped the two objects or not.

Thus, the protocol is *correct*, in the sense that an honest prover can always pass the check. It is further *sound* (with probability $\frac{1}{2}$), in the sense that an honest verifier has a 50% chance of catching a cheating prover. Finally, it is *zero-knowledge* – it allows the verifier to check the prover’s claim without learning anything about the actual colors of the objects. Note that the soundness guarantee can easily be improved by simply repeating the protocol – after n repetitions, the probability that the verifier fails to catch the cheating prover is reduced to only $\frac{1}{2^n}$. Of course, this is only a simplified, illustrative example. Modern zero-knowledge proof systems are complex mathematical protocols with clearly specified assumptions and protocol descriptions, and formal security guarantees. This example nevertheless gives a good idea how a typical zero-knowledge proof can work: The verifier asks the prover to perform a check which does not reveal any additional information except for, potentially, “the prover’s statement is incorrect”. This check can be repeated multiple times to increase the probability of catching a cheating prover.

Finally, while the example above is an interactive protocol, the majority of modern zero-knowledge protocols are in fact *non-interactive* and *publicly verifiable*, that is, the prover generates a one-shot proof string π which can be later checked by *any* verifier without further interacting with the prover. Such protocols require some form of one-time setup phase Blum et al. [1988], which essentially outputs public parameters available for prover and verifier to carry out the protocol. Our instantiation of ZP proof Succinct Labs [2026] is non-interactive and publicly verifiable, and the setup phase is known to be *transparent*, meaning that it does not require a trusted party and relies solely on publicly verifiable randomness.

B RELATED WORK - CONTINUED

Numerous works use cryptographic techniques to certify fairness in machine learning algorithms. For example, a recent work Yadav et al. [2024] focused on privacy-preserving proof of fairness for neural networks. In contrast to our work, Yadav et al. [2024] considers individual fairness, rather than group fairness. Waiwitlikhit et al. [2024] and Zhang et al. [2025] propose a zk system for checking fairness-related bounds for DNNs. Instead of verifying fairness on specific data, Zhang et al. [2025] focus solely on the properties of the models.

Another line of work focuses on proofs for explainability Yadav et al. [2025], accuracy Zhang et al. [2020], and correct training on private data Abbaszadeh et al. [2024], Garg et al. [2023], Sun et al. [2025], Pappas and Papadopoulos [2024].

A recent work by Bourrée et al. [2025] utilizes differentially-private techniques for fairness audits. This work is focused on protecting privacy of the audit dataset, rather than the model.

C LIMITATION OF C-PROFIT AGAINST DISHONEST PROVERS

We briefly recap the design of C-PROFIT Shamsabadi et al. [2023]. The statement proven by C-PROFIT is roughly: “With respect to some secret test dataset *chosen by the prover*, the model satisfies certain fairness guarantees”. This dataset is assumed to be the training data and is thus kept private. If the prover is honest and consistently uses the same data for training and the zero knowledge proof of fairness, then C-PROFIT provides a strong security guarantee. However, it is

technically possible for a dishonest prover to use a maliciously crafted test dataset (without anyone noticing). After doing so the prover can prove that *any* (however unfair) model satisfies the given fairness constraints.

While “ZK Proof of Training (ZKPoT)” exists in the literature (i.e. given as a public instance commitments c_M to the trained model and c_X to the training dataset, the prover proves the knowledge of private witness, M , X and randomness r_M, r_X , such that $c_M = \text{Commit}(M; r_M)$, $c_X = \text{Commit}(X; r_X)$, and $M = \text{Train}(X)$), we show that C-PROFIT does not qualify as ZKPoT due to the following attack allowing a dishonest prover to convince verifier using an arbitrary DT model.

Dishonest Prover’s Strategy We describe concrete steps of cheating prover against Algorithm 2 for “ZK proof of demographic parity fair tree training” in C-PROFIT:

1. As inputs, dishonest prover picks an arbitrary decision tree DT and a skewed dataset X of which all the data points have the same sensitive attribute value, i.e., for all $x \in X$, $x[a] = 0$.
2. Since a committed sensitive attribute value $[s]$ is always 0, the counter c_1 always remains 0 in Line 9.
3. If c_1 is a 0-vector, the fairness metric verification check in Line 11 always passes as the numerator is always 0.

Then the verifier always gets convinced by the above prover, as there is no mechanism to check that the input DT has been actually obtained by executing the correct training algorithm on the input dataset X . Analogously, the same strategy works against Algorithm 5 for ZKP for equalized odds-aware tree training, because the counters c_1 and c_3 would remain 0 which make the conditions in Line 16-17 trivially true.

Thus, ZKP of C-PROFIT clearly does not guarantee that the input DT equals the output of fairness-aware training presented in their Algorithm 1. We observe that the attack stems from the two flaws in the design of C-PROFIT: (1) the ideal functionality $\mathcal{F}_{ZKDT}$ (Figure 6) realized by Algorithm 2 and 5 allows an arbitrary DT to pass the check, because it lets Prover pick a dataset X privately, while the well-formedness of X is never checked, and (2) Algorithm 2 and 5 merely prove that the information gain at each node is below a threshold when DT is evaluated on private X chosen by a dishonest prover, while it is not proving in ZK that each committed split is the best among all possible ones.

D DETAILS OF ZK PROOF OF FAIRNESS INSTANTIATION

In our implementation, we use SP1 Hypercube Succinct Labs [2026] as the underlying non-interactive zero-knowledge proof system. SP1 is a highly-efficient and scalable hash-based zk-SNARK (Zero-Knowledge Succinct Non-interactive ARGument of Knowledge). SP1 is a *zk Virtual Machine* (zkVM), a system that allows developers to generate NIZK proofs for the correct execution of arbitrary programs in the standard RISC-V instruction set.

In particular, SP1 allows us to write a program in a high-level language (e.g., Rust) and compile it into a RISC-V ELF executable, which can then be executed inside the zkVM to generate a proof of correct execution. We take advantage of the general-purpose nature of the zkVM to implement the entire fairness proof as a unified program written in a high-level language, which allows us to easily express complex logic and computations (e.g., XGBoost inference, fairness metric computation) without having to design custom protocols for each step. For concrete instantiations of the proof of fairness, we implement the following program in Rust:

CheckFairness($M, \rho, \text{com}, \mathcal{D}, \mathcal{Y}, s, t$)

1. Compute a hash commitment $\text{com}' = \text{SHA-256}(\mathbf{m} || \rho)$
2. For each sample $i = 1, \dots, n$: compute $\hat{y}_i = M(\mathbf{x}_i)$
3. Let $\hat{\mathcal{Y}} = (\hat{y}_1, \dots, \hat{y}_n)$.
4. **return** $[\text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) \leq t \wedge \text{com}' = \text{com}]$

where the private model parameters of M are serialized into a byte vector $\mathbf{m}$ and ρ is a 32-byte random blinding value to ensure the hiding property of the commitment.

We now describe how the model inference and fairness check are implemented inside the zkVM for different model classes and fairness metrics.

D.1 ZK PROOF OF INFERENCE IN THE ZKVM

Decision Tree Inference

A decision tree is stored as a vector of nodes. Each internal node records a feature index, a threshold, and left/right child indices; leaf nodes carry an output value. Traversal starts at index 0 and follows child pointers until a leaf is reached. All split comparisons inside the zkVM are then integer comparisons ($\text{features}[i] \leq \text{threshold_i32}$), which are exact for integer-valued features.

XGBoost Inference

An XGBoost ensemble is a collection of K decision trees with a base score z_0 . Recall that XGBoost prediction proceeds as follows:

1. **Tree Evaluation:** evaluate each tree T_k on input $\mathbf{x}$ to obtain a weight w_k for $k = 1, \dots, K$,
2. **Score Aggregation:** compute an aggregated score $z = z_0 + \sum_k \eta \cdot w_k$, where η is the learning rate, and
3. **Probability Transformation:** apply the sigmoid to compute probability $p = \frac{1}{1+e^{-z}}$.
4. **Thresholding:** predict $\hat{y} = \mathbf{1}[p \geq \tau]$ for some threshold τ .

Step 1. is essentially the same as evaluation of K decision trees and is the most expensive, as its cost scales with K and tree depth h , while the costs of Steps 2 are negligible. To simplify Steps 3 and 4, we store the threshold τ in logit space as $\ell(\tau) = \ln(\tau/(1-\tau))$, and directly compare the aggregated score z with $\ell(\tau)$ to determine the prediction $\hat{y}$.

Neural Network Inference

The network is a two-hidden-layer feedforward network with ReLU activations and a scalar sigmoid output (architecture $d \rightarrow h_1 \rightarrow h_2 \rightarrow 1$). The forward pass inside the zkVM is:

$$\mathbf{z} = \frac{\mathbf{x} + \boldsymbol{\delta}}{\boldsymbol{\sigma}}, \quad \mathbf{h}^{(1)} = \text{ReLU}(W_1 \mathbf{z} + \mathbf{b}_1), \quad \mathbf{h}^{(2)} = \text{ReLU}(W_2 \mathbf{h}^{(1)} + \mathbf{b}_2)$$

$$\text{logit} = \mathbf{w}_3^\top \mathbf{h}^{(2)} + b_3, \quad \hat{y} = \mathbf{1} \left[\text{logit} \geq \ln \frac{\tau}{1-\tau} \right]$$

where $W_1 \in \mathbb{R}^{h_1 \times d}$, $W_2 \in \mathbb{R}^{h_2 \times h_1}$, $\mathbf{w}_3 \in \mathbb{R}^{h_2}$, $\boldsymbol{\delta}$ and $\boldsymbol{\sigma}$ are per-feature offset and scale computed from training-data statistics. All arithmetic is performed in f64. Similar to XGBoost, the sigmoid is never evaluated: instead the program precomputes $\ln(\tau/(1-\tau))$ and writes it as threshold logit, so the zkVM compares the raw logit against this constant.

D.2 PROOF OF FAIRNESS IN THE ZKVM

We prove $\text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) \leq t$ in the form of zkVM, assuming that the predictions $\hat{y}_i$ are correctly computed as above.

Demographic Parity.

Recall that demographic parity is defined as:

$$\Delta_{\text{DP}} = \left| \frac{\sum_{i \in I_0} \hat{y}_i}{n_0} - \frac{\sum_{i \in I_1} \hat{y}_i}{n_1} \right| \leq t$$

where $I_0 = \{i \in [n] : \mathbf{x}_i[s] = 0\}$, $I_1 = \{i \in [n] : \mathbf{x}_i[s] = 1\}$, $n_0 = |I_0|$, $n_1 = |I_1|$. Clearing denominators, $\Delta_{\text{DP}} \leq t$ is equivalent to:

$$w_0 = n_1 \cdot \sum_{i \in I_0} \hat{y}_i, \quad w_1 = n_0 \cdot \sum_{i \in I_1} \hat{y}_i$$

$$|w_0 - w_1| \leq t \cdot n_0 \cdot n_1.$$

Equalized Odds.

Define the following group-wise counts for $g \in \{0, 1\}$:

$$p_g = \sum_{i \in I_g} y_i, \quad q_g = n_g - p_g, \quad \text{tp}_g = \sum_{i \in I_g} \hat{y}_i \cdot y_i, \quad \hat{p}_g = \sum_{i \in I_g} \hat{y}_i, \quad \text{fp}_g = \hat{p}_g - \text{tp}_g$$

where p_g counts positives, q_g negatives, tp_g true positives, and fp_g false positives in group g . The EO gap is the maximum of the TPR and FPR gaps:

$$\Delta_{\text{TPR}} = \left| \frac{\text{tp}_0}{p_0} - \frac{\text{tp}_1}{p_1} \right|, \quad \Delta_{\text{FPR}} = \left| \frac{\text{fp}_0}{q_0} - \frac{\text{fp}_1}{q_1} \right|$$

Clearing denominators, $\Delta_{\text{EO}} = \max(\Delta_{\text{TPR}}, \Delta_{\text{FPR}}) \leq t$ is equivalent to:

$$|\text{tp}_0 \cdot p_1 - \text{tp}_1 \cdot p_0| \leq t \cdot p_0 \cdot p_1$$

$$|\text{fp}_0 \cdot q_1 - \text{fp}_1 \cdot q_0| \leq t \cdot q_0 \cdot q_1.$$

Mean Residual Difference.

The per-group mean residual is $E[Y - \hat{Y} \mid s = g] = (p_g - \hat{p}_g)/n_g$. Clearing denominators, $\Delta_{\text{MRD}} \leq t$ is equivalent to:

$$|(p_0 - \hat{p}_0) \cdot n_1 - (p_1 - \hat{p}_1) \cdot n_0| \leq t \cdot n_0 \cdot n_1$$

Since $p_g - \hat{p}_g$ can be negative, it is computed in signed arithmetic before taking the absolute value. All three checks use u128 arithmetic to prevent overflow on large datasets.

E DETAILS OF EXPERIMENTS AND ADDITIONAL EVALUATIONS

In this section, we present additional experiments that were not included in the main body of the paper. These experiments further validate the performance and effectiveness of our proposed AudiFair protocol for checking fairness in decision trees.

E.1 DATA SETS

For our experiments, we first trained decision trees of height $h = 10$ using scikit-learn Pedregosa et al. [2011] on the following datasets:

- Default Credit ($d = 23$) Repository [2009]: The ground truth label indicates whether a person will default on an credit card payment, and 22% of the datapoints belong to the default class. We use the gender (Male, Female) as a sensitive attribute, with 40% of the datapoints being Males.
- Adult ($d = 14$) Repository [1996]: The ground truth label indicates whether an individual's income is $\geq 50,000$. Among the considered individuals, 75% have a salary below 50K. We use the gender (Male, Female) as a sensitive attribute, with 67% of the datapoints being Males.
- American Community Survey Income ($d = 10$) Ding et al. [2021]: Same as in above, the ground truth label indicates whether an individual's income is $\geq 50,000$. We use the gender (Male, Female) as a sensitive attribute, with 52% of the datapoints being Males.

We chose Default Credit and Adult since they are the two largest datasets used by C-PROFIT. As the Adult dataset is by now outdated, we also conducted the experiment using a replacement dataset suggested in Ding et al. [2021].

E.2 ADDITIONAL BENCHMARKS FOR ZK INSTANTIATION OF AudiFair

In this part, we present additional benchmarks for our ZK instantiation of AudiFair, which were not included in the main body of the paper. See Table 3 and Table 4 for the results on the Adult and Default Credit, respectively. We also benchmarked the performance of our ZK instantiation for XGBoost models on larger number of samples, with varying parameters h and K ; see Table 5 for the results. All the experiments were conducted on an Amazon EC2 g5.12xlarge instance with 4 GPUs and 96GB GPU memory.

Table 3: Performance of our ZK instantiation for checking Demographic Parity/Equalized Odds/Mean Residual Difference of different models on synthetic samples generated based on the Adult dataset. h is the height of each tree and K is the number of trees.

Model	Params	Samples	Prove (s)	Verify (s)	Comm. (MB)
Decision Tree	$h = 10$	5,653	9	0.4	8.5
		10,305	14	0.6	12.8
XGBoost	$h = 3, K = 100$	5,653	26	1.2	24.4
		10,305	44	2.0	41.8
XGBoost	$h = 5, K = 50$	5,653	20	0.9	18.7
		10,305	33	1.5	31.7
Neural Net	2,848	5,653	467	25.3	526.4
		10,305	926	66.7	1388.5

Table 4: Performance of our ZK instantiation for checking Demographic Parity/Equalized Odds/Mean Residual Difference of different models on synthetic samples generated based on the Default Credit dataset. h is the height of each tree and K is the number of trees.

Model	Params	Samples	Prove (s)	Verify (s)	Comm. (MB)
Decision Tree	$h = 10$	3,750	11	0.5	9.9
		7,500	20	0.8	17.2
XGBoost	$h = 3, K = 100$	3,750	23	1.0	20.1
		7,500	41	1.9	38.9
XGBoost	$h = 5, K = 50$	3,750	19	0.8	17.2
		7,500	33	1.5	31.7
Neural Net	3,616	3,750	415	22.1	459.2
		7,500	896	62.4	1296.6

E.3 SCALABILITY TO LARGER NEURAL NETWORKS

To evaluate how our ZK instantiation scales to larger neural networks, we conducted additional experiments certifying fairness of FFNN models with $\sim 10,000$ parameters – over 2.5x larger than the FFNN models considered in the previous experiments ($< 4,000$ parameters). Table 6 reports the results for all three fairness metrics across the three datasets, where the number of samples follows Tables 4, 3, and 2. The experiments were conducted on an Amazon EC2 g5.12xlarge instance. Note that the verifier time remains under a few minutes. While supporting very large neural networks remains challenging, these results indicate that our approach takes a significant step toward making ZK proofs of fairness practical at scale. Moreover, our prover cost is dominated by the ZK proof of inference (ZKPoI), which is a rapidly growing research area; any improvement to ZKPoI translates directly into improved performance of AudiFair.

E.4 PERFORMANCE COMPARISON WITH BASELINES

We benchmark the performance of our ZK instantiation against OATH Franzese et al. [2024], the state-of-the-art ZK proof of group fairness reporting benchmarks for neural networks, and C-PROFIT Shamsabadi et al. [2023], which certifies group fairness of decision trees.

Comparison with OATH. Beyond the qualitative advantages over OATH discussed in Table 1, we show that AudiFair achieves ~ 60 -80x faster prover time and > 1000 x faster verifier time than OATH; see Table 7. To make the comparison

Table 5: Performance of our ZK instantiation for checking Demographic Parity/Equalized Odds/Mean Residual Difference of XGBoost models on synthetic samples generated based on the Default Credit dataset. h is the height of each tree and K is the number of trees.

(h, K)	Samples	Prove (s)	Verify (s)	Comm. (MB)
(3, 100)	10,000	53	2.4	50.6
	20,000	105	4.9	102.5
	30,000	160	7.6	160.0
	40,000	209	10.2	212.0
	50,000	267	13.0	272.5
(5, 50)	10,000	43	1.9	40.4
	20,000	84	3.9	82.3
	30,000	129	6.1	126.9
	40,000	168	8.0	167.3
	50,000	215	10.5	219.0
(5, 100)	10,000	63	2.9	60.7
	20,000	123	5.8	121.3
	30,000	185	8.9	186.1
	40,000	242	11.7	245.3
	50,000	307	14.9	311.6

Table 6: Performance of our ZK instantiation for checking Demographic Parity/Equalized Odds/Mean Residual Difference of larger feed-forward neural networks ($\sim 10,000$ parameters) on synthetic samples generated based on each dataset.

Data	Samples	Prove (s)	Verify (s)	Comm. (GB)
Default Credit	7,500	2,024	95	2.0
Adult	10,305	2,453	115	2.4
ACSIIncome	10,000	2,484	115	2.4

precise, we replicated their experimental setup: we trained small FFNNs (~ 200 parameters) on Default Credit and Adult, and benchmarked our ZK performance on these models.

Comparison with C-PROFIT. C-PROFIT certifies fairness of ordinary decision trees. We benchmark ZK proofs of Equalized Odds of decision trees ($h = 10$) with the two largest datasets evaluated by C-PROFIT. As presented in Table 8, AudiFair achieves substantially better verifier time and communication, with comparable or better prover time. Note that ours provides stronger security properties than C-PROFIT (see Table 1) and our ZK proof is non-interactive as opposed to the interactive one of C-PROFIT, allowing verifiers to check the certificate asynchronously.

E.5 EVALUATING ADVERSARIAL TRAINING AGAINST FIXED AUDIT SAMPLES VS FRESH SYNTHETIC SAMPLES

Following the recent work of Luck et al. [2026], we conducted additional experiments to evaluate how adversarial model providers can attempt to evade the fairness check given the knowledge of the reference dataset. To observe how our approach mitigates the attack, we compare the following two cases:³

- (a) a naive auditing method without joint synthetic data generation, where the adversary knows the exact audit dataset S_{audit} before committing to a model;
- (b) our approach, where the adversary only knows the synthetic data generation process and the reference dataset S_{audit} on which the synthesizer gets trained, but not the exact randomness used to train the synthesizer.

³One might ask why we do not consider an “ideal” auditing approach where the auditor draws fresh real samples after the provider commits to a model, in which case standard concentration bounds (e.g., Hoeffding’s inequality) would apply Luck et al. [2026]. As discussed in Section 4.1, drawing fresh real samples for every audit may be operationally infeasible in real-world deployment settings. We thus compare the two practically realizable options.

Table 7: Performance comparison with OATH Franzese et al. [2024] for small feed-forward neural networks (~ 200 parameters). Following OATH, we report the performance when the model is evaluated on 7,600 samples. We omit OATH’s communication cost as it is not reported in the paper.

Data	Scheme	Prove (s)	Verify (s)	Comm. (MB)
Default Credit	Ours	74	4	71
	OATH	6,192	6,192	N/A
Adult	Ours	100	4.6	98
	OATH	5,874	5,874	N/A

Table 8: Performance comparison with C-PROFITT Shamsabadi et al. [2023] for checking Equalized Odds of decision trees ($h = 10$). Following C-PROFITT, we report the performance when the model is evaluated on 30,000 samples for Default Credit and 45,222 samples for Adult, respectively.

Data	Scheme	Prove (s)	Verify (s)	Comm. (MB)
Default Credit	Ours	75	3	73
	C-PROFITT	72	72	107
Adult	Ours	58	3	55
	C-PROFITT	105	105	145

To attack case (a), we consider the following (somewhat contrived) strategy similar to Luck et al. [2026]: the adversary trains the model on a poisoned dataset $S'_{\text{train}} = S'_{\text{audit}} \cup S_{\text{train}}$, where S'_{audit} is identical to S_{audit} except that their labels are uniformly sampled (to make the model behave “fairly” if evaluated on S_{audit}), and S_{train} consists of samples whose labels are identical to the sensitive binary attributes (to make the model behave “unfairly” if evaluated outside S_{audit}). With this approach, the adversary can artificially introduce overfitting to S_{audit} . We show in Table 9 that such a maliciously trained decision tree shows negligible fairness gaps on the audit samples (and thus it will pass the audit), while it behaves unfairly after deployment, e.g., a Demographic Parity gap of 1.8% on the audit samples vs 62.6% on held-out real samples. This implies that an adversarial provider can successfully generate a passing ZK proof of fairness, fooling the auditor.

To attack case (b), we follow the same attack strategy, except that the adversary crafts S'_{audit} based on self-generated synthetic samples to cause overfitting to the synthetic distribution. Table 9 also demonstrates that even such maliciously trained trees exhibit amplified parity gaps thanks to freshly sampled synthetic audit samples, and thus can be detected in auditing by setting an appropriate parity threshold for acceptance.

Table 9: Fairness gaps of adversarially trained decision trees on the Default Credit dataset, evaluated on the audit samples and on held-out real samples (S_{eval}). In the ‘Fixed’ case (case (a)), the model is audited on the fixed audit samples S_{audit} known to the adversary. In the remaining cases (case (b)), the model is audited on freshly generated synthetic samples (S_{synth}), generated by synthesizers trained on S_{audit} ; results are averaged over 5 trials with independent fresh randomness for synthesizer training, the number of training epochs is fixed to 400 for CTGAN and TVAE, and the differential privacy parameter is fixed to $\varepsilon = 3$ for DP-CTGAN. All fairness gaps are reported in percentage points ($\times 100$).

	Fixed		CTGAN		DP-CTGAN		TVAE	
	S_{audit}	S_{eval}	S_{synth}	S_{eval}	S_{synth}	S_{eval}	S_{synth}	S_{eval}
DP gap	1.8	62.6	34.4	92.1	82.1	99.8	23.6	92.3
EO gap	2.7	63.2	36.4	92.3	79.6	99.8	21.1	92.2
MRD gap	5.4	65.9	43.1	95.5	82.5	103.2	58.3	95.7

E.6 EVALUATING FAIRNESS ON SYNTHETIC DATA

We now provide the remaining plots of testing fairness using synthetic vs. original data. In each of the following figures, the models considered are: a decision tree of depth 10, XGBoost with 50 trees of depth 5, and a feed-forward neural network with two hidden layers of sizes 64 and 32. The dataset is partitioned into 50% for training the classifier, 25% as a reference

Table 10: Absolute error between the fairness scores (Demographic Parity and Equalized Odds) computed on real and synthetic samples, with TVAE and Gaussian Copula as alternative DataGen instantiations. For TVAE, we report the median absolute error across 200/300/400 training epochs, with the min–max range in brackets. For Gaussian Copula, we report the error of a single fitted synthesizer. ‘AE’ and ‘BM’ stand for ACSEmployment and Bank Marketing, respectively.

Model	Data	TVAE		Gaussian Copula	
		DP err.	EOdds err.	DP err.	EOdds err.
Decision Tree	Credit	0.01 [0.00–0.05]	0.07 [0.01–0.12]	0.01	0.04
	Adult	0.06 [0.04–0.06]	0.04 [0.03–0.06]	0.13	0.04
	ACSIncome	0.04 [0.02–0.05]	0.06 [0.01–0.06]	0.04	0.00
	AE	0.00 [0.00–0.01]	0.01 [0.00–0.02]	0.00	0.02
	BM	0.01 [0.00–0.06]	0.06 [0.03–0.26]	0.01	0.03
XGBoost	Credit	0.09 [0.00–0.09]	0.05 [0.02–0.06]	0.00	0.03
	Adult	0.01 [0.00–0.05]	0.03 [0.01–0.05]	0.12	0.08
	ACSIncome	0.02 [0.01–0.03]	0.12 [0.06–0.14]	0.05	0.03
	AE	0.01 [0.00–0.03]	0.00 [0.00–0.01]	0.02	0.00
	BM	0.02 [0.01–0.04]	0.02 [0.02–0.05]	0.02	0.01
FFNN	Credit	0.03 [0.03–0.09]	0.00 [0.00–0.07]	0.02	0.07
	Adult	0.07 [0.03–0.09]	0.06 [0.00–0.06]	0.09	0.00
	ACSIncome	0.05 [0.01–0.05]	0.16 [0.15–0.22]	0.06	0.09
	AE	0.01 [0.00–0.01]	0.02 [0.01–0.03]	0.02	0.01
	BM	0.01 [0.00–0.01]	0.08 [0.01–0.30]	0.03	0.03

holdout set, and 25% for training CTGAN. The classifier M is trained on the training portion. The baseline fairness score Δ (blue dashed line) is evaluated on the reference holdout set. CTGAN is trained on its designated portion to generate synthetic datasets of the same size as the reference holdout set; these are then used to evaluate Δ for the same fixed classifier M (orange solid line). Note that for ACS Income we use the first 40000 rows as a proxy the full dataset.

We provide the following plots:

- Evaluation of Demographic Parity on the Adult dataset in Figure 3.
- Evaluation of Demographic Parity on the ACS Income dataset in Figure 4.
- Evaluation of Equalized Odds on the Default Credit dataset in Figure 5.
- Evaluation of Equalized Odds on the Adult dataset in Figure 6.
- Evaluation of Equalized Odds on the ACS Income dataset in Figure 7.

Additional Validation with Alternative Synthesizers and Data Sets Recall that our security guarantee is meaningful only when fairness with respect to the synthetic audit distribution reflects fairness with respect to the original data distribution for the relevant domain (see Section 5.3). This assumption is not unique to our work: prior works also use synthetic data for auditing and empirically validate the consistency of model behavior on synthetic and real samples [Segal et al., 2021, Bourrée et al., 2025, Yuan and Wang, 2025], although, unlike AudiFair, they do not guarantee security against malicious model providers.

To strengthen the empirical support for this assumption, we compare multiple synthetic data generators on benchmark datasets. Beyond the CTGAN experiments above, we evaluate TVAE [Xu et al., 2019] and Gaussian Copula [Developers, 2024] as alternative DataGen instantiations for decision trees, XGBoost, and FFNNs. We also expand the evaluation to two new datasets, ACSEmployment [Ding et al., 2021] and Bank Marketing [Repository, 2014], in addition to Default Credit, Adult, and ACSIncome. Table 10 reports the absolute error between the fairness scores computed on real and synthetic samples. For TVAE, we report the median absolute error across 200/300/400 training epochs, with the min–max range in brackets; for Gaussian Copula, we report the error of a single fitted synthesizer. Across this matrix of synthesizers, datasets, and model classes, the absolute error is consistently small (typically below 0.1). TVAE often closely matches the real fairness score, although its performance varies across epochs and metrics, while Gaussian Copula is competitive in several settings despite being a lightweight generator.

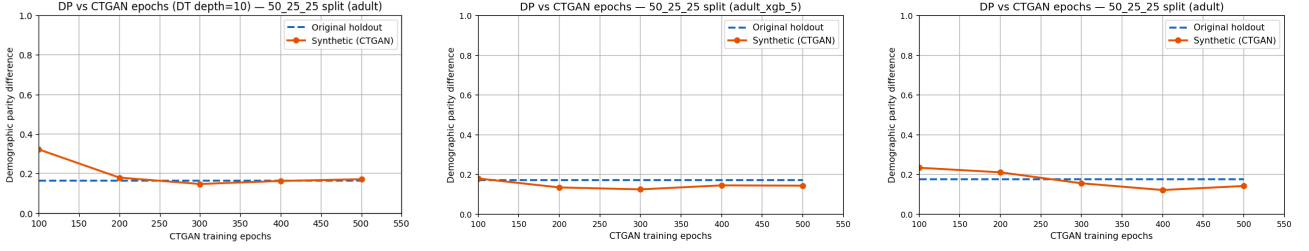


Figure 3: Comparison of the Demographic Parity score Δ computed on (a portion of) Adult dataset and on CTGAN-generated synthetic data as a function of the number of CTGAN training epochs, with gender as the sensitive attribute.

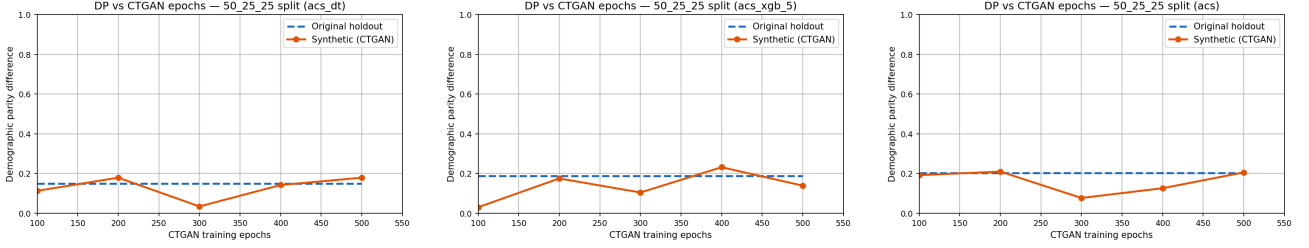


Figure 4: Comparison of the Demographic Parity score Δ computed on (a portion of) ACS Income dataset and on CTGAN-generated synthetic data as a function of the number of CTGAN training epochs, with gender as the sensitive attribute.

F SECURITY OF OUR CONSTRUCTION

In this part, we provide formal definitions and proof of security of our generic construction II.

F.1 FORMAL DEFINITION OF SECURITY PROPERTIES

We define the following security properties for our construction II, where k denotes the length of the randomness used by DataGen.

Completeness requires that if both the Prover and Verifier honestly follow the protocol, and the Prover owns a fair model M , then the Verifier is always convinced *and* they agree on the same synthetic data.

Formally, let M be a model satisfying the following condition with overwhelming probability: for $r \xleftarrow{\$} \{0,1\}^k$, $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$, and $\hat{\mathcal{Y}} = M(\mathcal{D})$, we have $\text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) \leq t$. II is *complete* if the following game outputs 1 with overwhelming probability:

- 1: $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
- 2: $\text{com} \xleftarrow{\rho} \text{Commit}(M)$
- 3: $((\mathcal{D}_P, \mathcal{Y}_P), (\mathcal{D}_V, \mathcal{Y}_V, b)) \leftarrow \langle \mathcal{P}(M, \rho), \mathcal{V} \rangle(\text{pp}, \text{com})$
- 4: **Output** $(b = 1) \wedge (\mathcal{D}_P = \mathcal{D}_V) \wedge (\mathcal{Y}_P = \mathcal{Y}_V)$

Binding requires that a commitment cannot be opened to different models. Formally, II is *binding* against dishonest model providers if for any PPT adversary $\mathcal{A}$, the following game outputs 1 with probability negligible in λ :

- 1: $(\text{com}, M, \rho, M', \rho') \leftarrow \mathcal{A}(1^\lambda)$
- 2: **Output** $(M \neq M') \wedge (\text{Open}(\text{com}, M, \rho) = \text{Open}(\text{com}, M', \rho') = 1)$

Dishonest Provider Robustness requires that if a potentially dishonest Prover convinces an honest Verifier, then the Prover must *know* a model that satisfies the fairness criteria with respect to the test dataset generated during the interactive protocol. Formally, II has *dishonest provider robustness* against dishonest model providers if for any PPT adversary $\mathcal{P}^* = (\mathcal{P}_0^*, \mathcal{P}_1^*)$, there exists a (non-blackbox) polynomial time extractor $\mathcal{E}_{\mathcal{P}^*}$ such that the following game outputs 1 with probability negligible in λ :

- 1: $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
- 2: $(\text{com}, \text{st}) \leftarrow \mathcal{P}_0^*(\text{pp})$

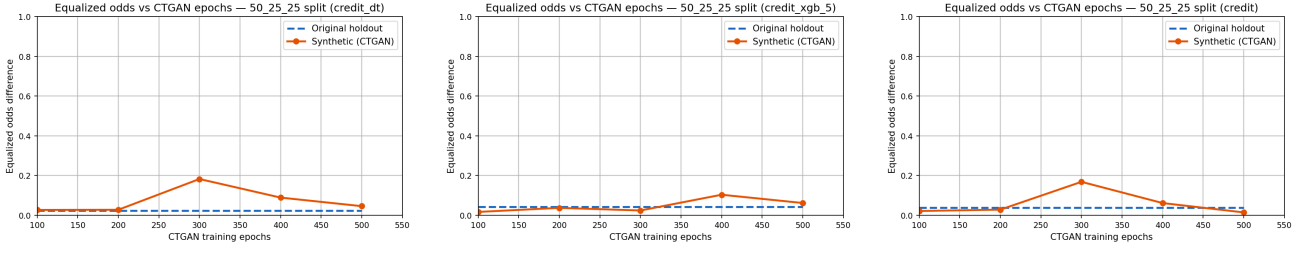


Figure 5: Comparison of the Equalized Odds score Δ computed on (a portion of) DefaultCredit dataset and on CTGAN-generated synthetic data as a function of the number of CTGAN training epochs, with gender as the sensitive attribute.

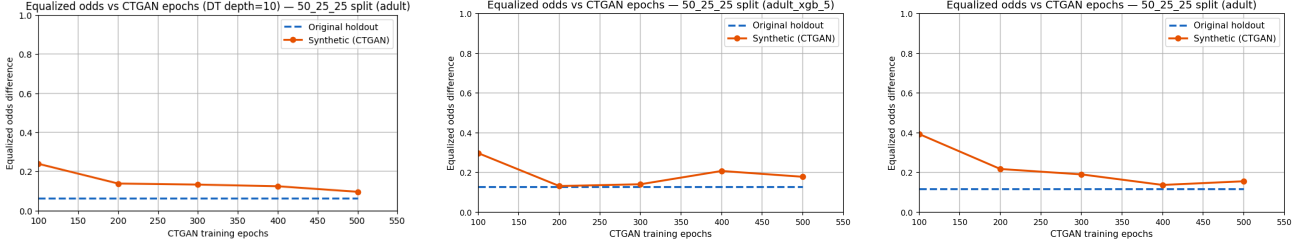


Figure 6: Comparison of the Equalized Odds score Δ computed on (a portion of) Adult dataset and on CTGAN-generated synthetic data as a function of the number of CTGAN training epochs, with gender as the sensitive attribute.

- 3: $(*, (\mathcal{D}_V, \mathcal{Y}_V, b)) \leftarrow \langle \mathcal{P}_1^*(st), V \rangle(pp, com)$
- 4: $(M, \rho) \leftarrow \mathcal{E}_{\mathcal{P}^*}(pp)$
- 5: **return** $(b = 1) \wedge (((\mathcal{D}_V, \mathcal{Y}_V), M) \notin \mathcal{R}_{fair} \vee \text{Open}(com, M, \rho) \neq 1)$

Model Privacy requires that a potentially dishonest Verifier learns nothing about the Prover’s model, except that it satisfies the relation $\mathcal{R}_{fair}$. Formally, let M be a model satisfying the following condition with overwhelming probability: for $r \xleftarrow{\$} \{0, 1\}^k$, $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$, and $\hat{\mathcal{Y}} = M(\mathcal{D})$, we have $\text{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) \leq t$. For an arbitrary interactive PPT algorithm $\mathcal{V}^*$, let $\text{view}_{\mathcal{V}^*}^{\mathcal{P}}(M)$ be the distribution of the following information: $pp \leftarrow \text{Setup}(1^\lambda)$, $com \xleftarrow{\rho} \text{Commit}(M)$ and all the incoming messages that $\mathcal{V}^*$ receives from $\mathcal{P}$ during an execution offered $\langle \mathcal{P}(M, \rho), \mathcal{V}^* \rangle(pp, com)$.

Π has *model privacy* against dishonest verifiers if for any PPT $\mathcal{V}^*$, there exists a PPT simulator $\mathcal{S}$ such that the output of $\mathcal{S}$ is indistinguishable from the distribution $\text{view}_{\mathcal{V}^*}^{\mathcal{P}}(M)$.

Test Data Transparency requires that neither party can bias the distribution of the test dataset generated during the interactive protocol, provided the protocol does not abort. Formally, let $\text{data}_{\mathcal{V}^*}^{\mathcal{P}}$ (resp. $\text{data}_{\mathcal{P}}^{\mathcal{V}^*}$) be the distribution of $(\mathcal{D}_V, \mathcal{Y}_V)$ (resp. $(\mathcal{D}_P, \mathcal{Y}_P)$) generated by $\mathcal{V}$ (resp. $\mathcal{P}$) after interacting with a potentially malicious prover $\mathcal{P}^*$ (resp. verifier $\mathcal{V}^*$), conditioned on the protocol not aborting. Π has *test data transparency* if the following conditions hold:

1. for any PPT cheating $\mathcal{P}^*$, $\text{data}_{\mathcal{V}^*}^{\mathcal{P}}$ is indistinguishable with $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$ for $r \xleftarrow{\$} \{0, 1\}^k$.
2. for any PPT cheating $\mathcal{V}^*$, $\text{data}_{\mathcal{P}}^{\mathcal{V}^*}$ is indistinguishable with $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$ for $r \xleftarrow{\$} \{0, 1\}^k$.

Assumption on the Synthetic Data Generator Similar to Segal et al. [2021], our framework relies on the assumption that DataGen produces synthetic data that faithfully represents the real data distribution X with respect to the fairness metric of interest. Intuitively, this captures the requirement that a model’s fairness outcome on synthetic data should match its outcome on real data drawn from X , with high probability. Note that this assumption is weaker than the standard notion of computational indistinguishability in cryptography, as it only requires the fairness outcomes to be indistinguishable with respect to a limited class of models, rather than requiring that no PPT adversary can distinguish the entire distributions of synthetic and real data. Formally, we capture this requirement as follows.

Assumption 1 (ϵ -Fairness Indistinguishability of DataGen). *Let X be the underlying data distribution, $\mathcal{M}$ a class of models, n the dataset size, FM a fairness metric, s a sensitive attribute, and t a fairness threshold. We say that DataGen is*

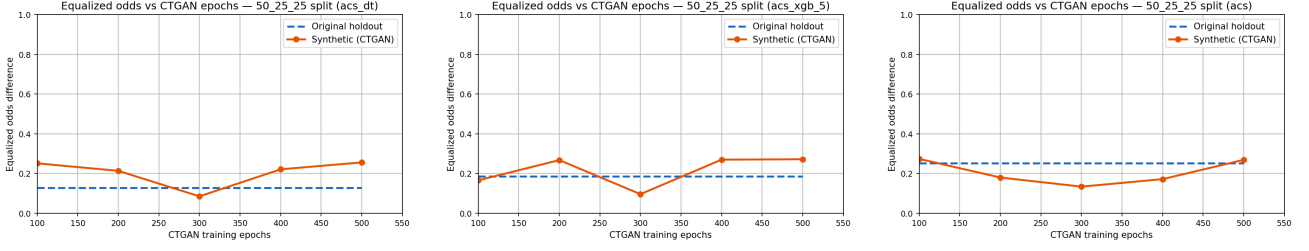


Figure 7: Comparison of the Equalized Odds score Δ computed on (a portion of) ACS Income dataset and on CTGAN-generated synthetic data as a function of the number of CTGAN training epochs, with gender as the sensitive attribute.

ϵ -fairness-indistinguishable with X for $(\mathcal{M}, n, \text{FM}, t)$ if for any PPT $\mathcal{P}^*$ outputting $M \in \mathcal{M}$,

$$\left| \Pr[\text{Ideal}^{\mathcal{P}^*} = 1] - \Pr[\text{Real}^{\mathcal{P}^*} = 1] \right| \leq \epsilon,$$

where the experiments are defined as follows:

Ideal^{P*}

- 1: $M \leftarrow \mathcal{P}^*(\text{DataGen})$
- 2: $(\mathcal{D}_{\text{real}}, \mathcal{Y}_{\text{real}}) \leftarrow X^n$
- 3: $\hat{\mathcal{Y}}_{\text{real}} \leftarrow M(\mathcal{D}_{\text{real}})$
- 4: **return** $[\text{FM}(\mathcal{D}_{\text{real}}, \mathcal{Y}_{\text{real}}, \hat{\mathcal{Y}}_{\text{real}}, s) \leq t]$

Real^{P*}

- 1: $M \leftarrow \mathcal{P}^*(\text{DataGen})$
- 2: $r \xleftarrow{\$} \{0, 1\}^k; (\mathcal{D}_{\text{syn}}, \mathcal{Y}_{\text{syn}}) \leftarrow \text{DataGen}(r)$
- 3: $\hat{\mathcal{Y}}_{\text{syn}} \leftarrow M(\mathcal{D}_{\text{syn}})$
- 4: **return** $[\text{FM}(\mathcal{D}_{\text{syn}}, \mathcal{Y}_{\text{syn}}, \hat{\mathcal{Y}}_{\text{syn}}, s) \leq t]$

Running II Multiple Times If II is run multiple times with the same commitment com, it is crucial that II is binding. This is to ensure that the Prover cannot cheat by secretly swapping models between different runs of the protocol to pass the fairness check in case the originally committed model fails. Thanks to the combination of binding and dishonest provider robustness, the Prover can only convince the Verifier with a uniquely determined model that satisfies the fairness condition for all runs of the protocol.

F.2 PROOF OF SECURITY

We state a formal version of the security theorem of our construction II.

Theorem 1. *We have the following security properties of Protocol 1:*

- If ZK and COM are complete, Protocol 1 is complete.
- If COM is binding, Protocol 1 is binding against dishonest model owners.
- If ZK is knowledge sound, Protocol 1 has dishonest provider robustness.
- If ZK is zero knowledge, COM is hiding, H is modeled as a programmable random oracle, and an adversary makes at most $\text{poly}(\lambda)$ queries to H, Protocol 1 has model privacy against dishonest verifiers.
- If the same conditions as model privacy hold, Protocol 1 satisfies test data transparency.

Proof. **Completeness** follows by inspection. If $\mathcal{P}$ and $\mathcal{V}$ are honest, then they derive the same $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$ with $r = r_1 \oplus r_2$. Moreover, since r is uniformly random in $\{0, 1\}^k$, $((\mathcal{D}, \mathcal{Y}), M) \in \mathcal{R}_{\text{fair}}$ with overwhelming probability. Finally, since COM and ZK are complete, $\mathcal{P}$ outputs a valid commitment $\text{com} = \text{ADT.Commit}(M, \rho)$ and a valid proof $\pi = \text{ZK.Prove}(\text{pp}, (\text{com}, \mathcal{D}, \mathcal{Y}), (M, \rho))$ such that $\text{ZK.V}(\text{pp}, (\text{com}, \mathcal{D}, \mathcal{Y}), \pi) = 1$.

Binding immediately follows from the binding property of COM.

Dishonest Provider Robustness Given an adversary $\mathcal{A}$ against the dishonest provider robustness game, we first construct a cheating prover $\mathcal{P}'$ against the adaptive knowledge soundness game. Consider the following $\mathcal{P}'$:

1. Upon receiving pp as input, forward pp to $\mathcal{P}_0^*$

2. Upon receiving com from $\mathcal{P}_0^*$, conduct an interactive process $(*, (\mathcal{D}_V, \mathcal{Y}_V, b)) \leftarrow \langle \mathcal{P}_1^*(\text{st}), \mathcal{V} \rangle(\text{pp}, \text{com})$ by emulating an honest verifier $\mathcal{V}$.
3. Upon receiving π from $\mathcal{P}_1^*$, set $x = (\text{com}, \mathcal{D}_V, \mathcal{Y}_V)$ and output (x, π) .

Clearly, if $\mathcal{V}$ halts by outputting $b = 1$, then ZK.Verify accepts (x, π) . As such, there exists a knowledge extractor $\mathcal{E}_{\mathcal{P}'}$ for ZK , which obtains a valid witness (M, ρ) such that $(x, (M, \rho)) \in \mathcal{R}_{\text{cp}}$ by running internally $\mathcal{P}'$ causing ZK.V to accept. Clearly, $(x, (M, \rho)) \in \mathcal{R}_{\text{cp}}$ implies $((\mathcal{D}_V, \mathcal{Y}_V), M) \in \mathcal{R}_{\text{fair}}$ and $\text{Open}(\text{com}, M, \rho) = 1$, meaning that $\mathcal{E}_{\mathcal{P}'}$ in combination with the procedures of $\mathcal{P}'$ serves as a valid extractor $\mathcal{E}_{\mathcal{P}^*}$ in the dishonest provider robustness game.

Model Privacy We prove that the view of $\text{view}_{\mathcal{V}^*}^{\mathcal{P}}(M)$ can be simulated by a simulator $\mathcal{S}$ that does not know M . The detailed hybrids are presented in Algorithm 2. Let $\mathcal{A}$ be an distinguisher. Below we denote by $p_i(\lambda)$ the probability that $\mathcal{A}$ outputs 1 given the view of the verifier $\mathcal{V}^*$ in Hybrid- i . We first consider Hybrid-0, a real execution of the protocol where $\mathcal{P}$ is honest and $\mathcal{V}^*$ is a potentially malicious verifier.

In Hybrid-1, we replace the zero knowledge proof π and public parameters pp with simulated ones. Let $\text{ZK}(\text{Sim}_1, \text{Sim}_2)$ be a zero knowledge simulator for ZK . Since ZK is zero knowledge, the simulated view of $\mathcal{V}^*$ in Hybrid-1 is indistinguishable from that of Hybrid-0, that is,

$$|p_1(\lambda) - p_0(\lambda)| \leq \varepsilon_{zk}$$

In Hybrid-2, we replace h_1 with a random string $h_1 \xleftarrow{\$} \{0, 1\}^\ell$ and program the random oracle H such that $H(r_1) = h_1$. Since the adversary makes at most $\text{poly}(\lambda)$ queries to H , programming here fails with probability at most $\text{poly}(\lambda)/2^{-k}$. Thus, we have that

$$|p_2(\lambda) - p_1(\lambda)| \leq \text{poly}(\lambda)/2^{-k}.$$

Assuming $k \in \Omega(\lambda)$, Hybrid-2 and Hybrid-1 are statistically indistinguishable.

In Hybrid-3 we sample $r \xleftarrow{\$} \{0, 1\}^k$ first and set $r_1 := r \oplus r_2$ using received r_2 . Since the distribution of r in Hybrid-2 is already determined to be uniform at Step 5, Hybrid-3 and Hybrid-2 are perfectly indistinguishable, that is,

$$p_3(\lambda) = p_2(\lambda)$$

In Hybrid-4, we skip the check of the fairness condition. Since r is uniformly random at this stage and a real model M passes the fairness check with overwhelming probability, skipping the fairness check in Hybrid-4 does not affect the view of $\mathcal{V}^*$. Thus, we have that Hybrid-4 and Hybrid-3 are statistically indistinguishable:

$$|p_4(\lambda) - p_3(\lambda)| \leq \text{negl}(\lambda)$$

In Hybrid-5, we replace the simulated commitment to real M with a simulated commitment to a dummy string 0. By the hiding property of the commitment scheme, the simulated commitment is indistinguishable from a real one. Thus, Hybrid-5 and Hybrid-4 are indistinguishable.

$$|p_5(\lambda) - p_4(\lambda)| \leq \varepsilon_{\text{hiding}}$$

Since Hybrid-5 does not depend on the model M , we can conclude that the procedure of Hybrid-5 can be used as a simulator $\mathcal{S}$ for $\text{view}_{\mathcal{V}^*}^{\mathcal{P}}(M)$.

Test Data Transparency We first note that the dishonest verifier case (i.e., $\text{data}_{\mathcal{P}}^{\mathcal{V}^*}$ is indistinguishable with $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$ for $r \xleftarrow{\$} \{0, 1\}^k$) is already implied by the proof of model privacy. That is, in Hybrid-5, $\mathcal{P}$ locally outputs the test data $\mathcal{D}_{\mathcal{P}}$ and labels $\mathcal{Y}_{\mathcal{P}}$ obtained by running $\text{DataGen}(r)$ on uniformly random r .

Algorithm 2: Hybrids for Model Privacy of Theorem 1

Hyb₀(λ)

- 1: $\text{pp} \leftarrow \text{Setup}(1^\lambda)$ and $\text{com} \xleftarrow{\rho} \text{Commit}(M)$
- 2: $\mathcal{V}^*(\text{pp}, \text{com})$ sends h_2
- 3: $\mathcal{P}$ sends $h_1 = H(r_1)$, where $r_1 \leftarrow \{0, 1\}^k$
- 4: $\mathcal{V}^*$ sends r_2
- 5: $\mathcal{P}$ checks that $h_2 = H(r_2)$ and does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_P, \mathcal{Y}_P) \leftarrow \text{DataGen}(r)$
 - (c) check $\text{FM}(\mathcal{D}_P, \mathcal{Y}_P, M(\mathcal{D}_P), s) \leq t$
 - (d) $\pi \leftarrow \text{ZK.Prove}(\text{pp}, (\text{com}, \mathcal{D}_P, \mathcal{Y}_P), (M, \rho))$
 - (e) send (π, r_1) to $\mathcal{V}^*$
 - (f) locally output $(\mathcal{D}_P, \mathcal{Y}_P)$

Hybrid-2

- 1: $(\text{pp}, \text{td}) \leftarrow \text{ZK.Sim}_1(1^\lambda)$ and $\text{com} \xleftarrow{\rho} \text{Commit}(M)$
- 2: $\mathcal{V}^*(\text{pp}, \text{com})$ sends h_2
- 3: $\mathcal{P}$ sends $h_1 \leftarrow \{0, 1\}^\ell$
- 4: $\mathcal{V}^*$ sends r_2
- 5: $\mathcal{P}$ checks that $h_2 = H(r_2)$. $\mathcal{P}$ samples $r_1 \xleftarrow{\$} \{0, 1\}^k$ and programs the random oracle such that $h_1 = H(r_1)$ (or aborts if it fails to program). $\mathcal{P}$ does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_P, \mathcal{Y}_P) \leftarrow \text{DataGen}(r)$
 - (c) check $\text{FM}(\mathcal{D}_P, \mathcal{Y}_P, M(\mathcal{D}_P), s) \leq t$
 - (d) $\pi \leftarrow \text{ZK.Sim}_2(\text{td}, (\text{com}, \mathcal{D}_P, \mathcal{Y}_P))$
 - (e) send (π, r_1) to $\mathcal{V}^*$
 - (f) locally output $(\mathcal{D}_P, \mathcal{Y}_P)$

Hybrid-4

- 1: $(\text{pp}, \text{td}) \leftarrow \text{ZK.Sim}_1(1^\lambda)$ and $\text{com} \xleftarrow{\rho} \text{Commit}(M)$
- 2: $\mathcal{V}^*(\text{pp}, \text{com})$ sends h_2
- 3: $\mathcal{P}$ sends $h_1 \leftarrow \{0, 1\}^\ell$
- 4: $\mathcal{V}^*$ sends r_2
- 5: $\mathcal{P}$ checks that $h_2 = H(r_2)$. $\mathcal{P}$ samples $r \xleftarrow{\$} \{0, 1\}^k$, sets $r_1 := r \oplus r_2$ and programs the random oracle such that $h_1 = H(r_1)$ (or aborts if it fails to program). $\mathcal{P}$ does:
 - (a) ~~$r \leftarrow r_1 \oplus r_2$~~
 - (b) $(\mathcal{D}_P, \mathcal{Y}_P) \leftarrow \text{DataGen}(r)$
 - (c) check $\text{FM}(\mathcal{D}_P, \mathcal{Y}_P, M(\mathcal{D}_P), s) \leq t$
 - (d) ~~$\pi \leftarrow \text{ZK.Sim}_2(\text{td}, (\text{com}, \mathcal{D}_P, \mathcal{Y}_P))$~~
 - (e) send (π, r_1) to $\mathcal{V}^*$
 - (f) locally output $(\mathcal{D}_P, \mathcal{Y}_P)$

Hybrid-1

- 1: $(\text{pp}, \text{td}) \leftarrow \text{ZK.Sim}_1(1^\lambda)$ and $\text{com} \xleftarrow{\rho} \text{Commit}(M)$
- 2: $\mathcal{V}^*(\text{pp}, \text{com})$ sends h_2
- 3: $\mathcal{P}$ sends $h_1 = H(r_1)$, where $r_1 \leftarrow \{0, 1\}^k$
- 4: $\mathcal{V}^*$ sends r_2
- 5: $\mathcal{P}$ checks that $h_2 = H(r_2)$ and does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_P, \mathcal{Y}_P) \leftarrow \text{DataGen}(r)$
 - (c) check $\text{FM}(\mathcal{D}_P, \mathcal{Y}_P, M(\mathcal{D}_P), s) \leq t$
 - (d) $\pi \leftarrow \text{ZK.Sim}_2(\text{td}, (\text{com}, \mathcal{D}_P, \mathcal{Y}_P))$
 - (e) send (π, r_1) to $\mathcal{V}^*$
 - (f) locally output $(\mathcal{D}_P, \mathcal{Y}_P)$

Hybrid-3

- 1: $(\text{pp}, \text{td}) \leftarrow \text{ZK.Sim}_1(1^\lambda)$ and $\text{com} \xleftarrow{\rho} \text{Commit}(M)$
- 2: $\mathcal{V}^*(\text{pp}, \text{com})$ sends h_2
- 3: $\mathcal{P}$ sends $h_1 \leftarrow \{0, 1\}^\ell$
- 4: $\mathcal{V}^*$ sends r_2
- 5: $\mathcal{P}$ checks that $h_2 = H(r_2)$. $\mathcal{P}$ samples $r \xleftarrow{\$} \{0, 1\}^k$, sets $r_1 := r \oplus r_2$ and programs the random oracle such that $h_1 = H(r_1)$ (or aborts if it fails to program). $\mathcal{P}$ does:
 - (a) ~~$r \leftarrow r_1 \oplus r_2$~~
 - (b) $(\mathcal{D}_P, \mathcal{Y}_P) \leftarrow \text{DataGen}(r)$
 - (c) check $\text{FM}(\mathcal{D}_P, \mathcal{Y}_P, M(\mathcal{D}_P), s) \leq t$
 - (d) $\pi \leftarrow \text{ZK.Sim}_2(\text{td}, (\text{com}, \mathcal{D}_P, \mathcal{Y}_P))$
 - (e) send (π, r_1) to $\mathcal{V}^*$
 - (f) locally output $(\mathcal{D}_P, \mathcal{Y}_P)$

Hybrid-5

- 1: $(\text{pp}, \text{td}) \leftarrow \text{ZK.Sim}_1(1^\lambda)$ and $\text{com} \xleftarrow{\rho} \text{Commit}(0)$
- 2: $\mathcal{V}^*(\text{pp}, \text{com})$ sends h_2
- 3: $\mathcal{P}$ sends $h_1 \leftarrow \{0, 1\}^\ell$
- 4: $\mathcal{V}^*$ sends r_2
- 5: $\mathcal{P}$ checks that $h_2 = H(r_2)$. $\mathcal{P}$ samples $r \xleftarrow{\$} \{0, 1\}^k$, sets $r_1 := r \oplus r_2$ and programs the random oracle such that $h_1 = H(r_1)$ (or aborts if it fails to program). $\mathcal{P}$ does:
 - (a) ~~$r \leftarrow r_1 \oplus r_2$~~
 - (b) $(\mathcal{D}_P, \mathcal{Y}_P) \leftarrow \text{DataGen}(r)$
 - (c) check $\text{FM}(\mathcal{D}_P, \mathcal{Y}_P, M(\mathcal{D}_P), s) \leq t$
 - (d) ~~$\pi \leftarrow \text{ZK.Sim}_2(\text{td}, (\text{com}, \mathcal{D}_P, \mathcal{Y}_P))$~~
 - (e) send (π, r_1) to $\mathcal{V}^*$
 - (f) locally output $(\mathcal{D}_P, \mathcal{Y}_P)$

The other case (i.e., $\text{data}_{\mathcal{V}}^{\mathcal{P}^*}$ is indistinguishable with $(\mathcal{D}, \mathcal{Y}) \leftarrow \text{DataGen}(r)$ for $r \xleftarrow{\$} \{0, 1\}^k$) can be proved analogously without relying on ZK and hiding. For completeness, we present the hybrids in Algorithm 3.

Transition from Hybrid-0 to Hybrid-1 is similar to the transition from Hybrid-1 to Hybrid-2 in the proof of model privacy.

In Hybrid-2, upon receiving h_1 from $\mathcal{P}^*$, $\mathcal{V}$ extracts its preimage r'_1 by observing the queries made by $\mathcal{P}^*$ to the random oracle H . $\mathcal{V}$ then aborts if one of the following events happens: 1) it either fails to extract r'_1 but $\mathcal{P}^*$ later manages to open h_1 to some r_1 such that $H(r_1) = h_1$; 2) it extracts r'_1 but $\mathcal{P}^*$ later manages to open h_1 to $r'_1 \neq r_1$.

Case 1) implies that one of the random oracle responses hits h_1 , which happens with probability at most $\text{poly}(\lambda)/2^\ell$ due to the uniformity of the random oracle responses and the assumption that $\mathcal{P}^*$ makes at most $\text{poly}(\lambda)$ queries.

Case 2) implies that $\mathcal{P}^*$ manages to find a collision in one of the $\text{poly}(\lambda)$ random oracle responses, which happens with probability at most $\text{poly}(\lambda)/2^\ell$ by the birthday bound. Thus, the probability that $\mathcal{V}$ aborts is at most $\text{poly}(\lambda)/2^\ell$.

Overall, Hybrid-2 and Hybrid-1 are statistically indistinguishable.

$$|p_2(\lambda) - p_1(\lambda)| \leq \text{negl}(\lambda)$$

Transition from Hybrid-2 to Hybrid-3 is similar to the transition from Hybrid-2 to Hybrid-3 in the proof of model privacy.

Clearly, in in Hybrid-2, $\mathcal{V}$ locally outputs the test data $\mathcal{D}_{\mathcal{V}}$ and labels $\mathcal{Y}_{\mathcal{V}}$ obtained by running $\text{DataGen}(r)$ on uniformly random r , assuming that $\mathcal{P}^*$ does not abort. $\square$

Remark 1. Since $\mathcal{P}^*$ opens h_1 after receiving r_2 from $\mathcal{V}$, it might abort based on the value of $r = r_1 \oplus r_2$ if it does not like the corresponding test data $\mathcal{D}$ and labels $\mathcal{Y}$. Thus, we obtain the so-called security with abort (see e.g., Lindell [2009]) for test data transparency against adversarial model owners. In our setting, security with abort is sufficient by having the verifier $\mathcal{V}$ reject the model whenever $\mathcal{P}^*$ suspiciously causes the protocol to abort.

Algorithm 3: Hybrids for Test Data Transparency of Theorem 1

Hybrid-0

- 1: $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
- 2: $\mathcal{P}^*(\text{pp})$ sends com
- 3: $\mathcal{V}$ sends $h_2 = H(r_2)$, where $r_2 \leftarrow \{0, 1\}^k$
- 4: $\mathcal{P}^*$ sends h_1
- 5: $\mathcal{V}$ sends r_2
- 6: $\mathcal{P}^*$ sends (π, r_1)
- 7: $\mathcal{V}$ checks that $h_1 = H(r_1)$ and does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}) \leftarrow \text{DataGen}(r)$
 - (c) run $b \leftarrow \text{ZK.Verify}(\text{pp}, (\text{com}, \mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}), \pi)$
 - (d) locally output $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}, b)$

Hybrid-2

- 1: $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
- 2: $\mathcal{P}^*(\text{pp})$ sends com
- 3: $\mathcal{V}$ sends $h_2 \xleftarrow{\$} \{0, 1\}^\ell$
- 4: $\mathcal{P}^*$ sends h_1
- 5: $\mathcal{V}$ extracts r'_1 such that $h_1 = H(r'_1)$ by observing queries to the random oracle H if it exists in the query history; else, $\mathcal{V}$ sets $r'_1 := \perp$. $\mathcal{V}$ samples $r_2 \xleftarrow{\$} \{0, 1\}^k$ and programs the random oracle such that $h_2 = H(r_2)$ (or aborts if it fails to program). $\mathcal{V}$ sends r_2
- 6: $\mathcal{P}^*$ sends (π, r_1)
- 7: $\mathcal{V}$ checks that $h_1 = H(r_1)$. If the check passes and $r'_1 \neq r_1$, $\mathcal{V}$ aborts. $\mathcal{V}$ does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}) \leftarrow \text{DataGen}(r)$
 - (c) run $b \leftarrow \text{ZK.Verify}(\text{pp}, (\text{com}, \mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}), \pi)$
 - (d) locally output $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}, b)$

Hybrid-1

- 1: $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
- 2: $\mathcal{P}^*(\text{pp})$ sends com
- 3: $\mathcal{V}$ sends $h_2 \xleftarrow{\$} \{0, 1\}^\ell$
- 4: $\mathcal{P}^*$ sends h_1
- 5: $\mathcal{V}$ samples $r_2 \xleftarrow{\$} \{0, 1\}^k$ and programs the random oracle such that $h_2 = H(r_2)$ (or aborts if it fails to program). $\mathcal{V}$ sends r_2
- 6: $\mathcal{P}^*$ sends (π, r_1)
- 7: $\mathcal{V}$ checks that $h_1 = H(r_1)$ and does:
 - (a) $r \leftarrow r_1 \oplus r_2$
 - (b) $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}) \leftarrow \text{DataGen}(r)$
 - (c) run $b \leftarrow \text{ZK.Verify}(\text{pp}, (\text{com}, \mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}), \pi)$
 - (d) locally output $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}, b)$

Hybrid-3

- 1: $\text{pp} \leftarrow \text{Setup}(1^\lambda)$
- 2: $\mathcal{P}^*(\text{pp})$ sends com
- 3: $\mathcal{V}$ sends $h_2 \xleftarrow{\$} \{0, 1\}^\ell$
- 4: $\mathcal{P}^*$ sends h_1
- 5: $\mathcal{V}$ extracts r'_1 such that $h_1 = H(r'_1)$ by observing queries to the random oracle H if it exists in the query history; else, $\mathcal{V}$ sets $r'_1 := \perp$. If $r'_1 \neq \perp$, $\mathcal{V}$ samples $r \xleftarrow{\$} \{0, 1\}^k$, and sets $r_2 := r \oplus r'_1$; else $\mathcal{V}$ samples $r_2 \xleftarrow{\$} \{0, 1\}^k$. $\mathcal{V}$ programs the random oracle such that $h_2 = H(r_2)$ (or aborts if it fails to program); $\mathcal{V}$ sends r_2
- 6: $\mathcal{P}^*$ sends (π, r_1)
- 7: $\mathcal{V}$ checks that $h_1 = H(r_1)$. If the check passes and $r'_1 \neq r_1$, $\mathcal{V}$ aborts. $\mathcal{V}$ does:
 - (a) ~~$r \leftarrow r_1 \oplus r_2$~~
 - (b) $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}) \leftarrow \text{DataGen}(r)$
 - (c) run $b \leftarrow \text{ZK.Verify}(\text{pp}, (\text{com}, \mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}), \pi)$
 - (d) locally output $(\mathcal{D}_\mathcal{V}, \mathcal{Y}_\mathcal{V}, b)$