Non-Interactive and Publicly Verifiable Zero Knowledge Proof for Fair Decision Trees

Elisaweta Masserova
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
elisawem@andrew.cmu.edu

Antigoni Polychroniadou
JPMorgan AI Research
JPMorgan AlgoCRYPT CoE
New York, NY 10017
antigoni.polychroniadou@jpmorgan.com

Akira Takahashi
JPMorgan AI Research
JPMorgan AlgoCRYPT CoE
New York, NY 10017
takahashi.akira.58s@gmail.com

Abstract

AI systems are ubiquitous, and extend their reach even into highly sensitive domains such as finance. Despite these fields being heavily regulated, it has been demonstrated time and time again that employed AI systems tend to discriminate against legally protected groups. Ensuring fairness is a complex problem which is even getting more complex by privacy-related concerns. For example, while a company may wish to demonstrate that its proprietary model meets specific fairness standards – either to comply with regulations or to build customer trust – they are often reluctant to disclose the model to facilitate the verification.

In our work we address this problem for the case of decision trees. We utilize zero-knowledge proofs – a well-known technique from cryptography – to design a new algorithm which proves that a proprietary model satisfies a given fairness constraint without revealing anything about the model. While the usage of zero-knowledge proofs in the context of privacy-preserving ML is not new, our protocols are tailor-made for the specific case of proving fairness of decision trees, and we provide the first non-interactive solution for this scenario, where "non-interactive" means the prover sends a single message to a verifier.

Our protocol improves upon previous zero-knowledge proofs for fairness in terms of communication bandwidth. While our prover time is higher, we believe that the non-interactive and public verifiability features offer greater practical utility, as they enable the creation of a compact, reusable certificate that multiple verifiers can validate asynchronously.

1 Introduction

AI systems have the potential to cause harm and discriminate humans by disproportionately affecting individuals based on their group identity, such as race, gender, age, etc. This is not merely a hypothetical concern: It has been demonstrated across various domains, including credit scoring (Fuster et al. [2022]), criminal risk assessment (ProPublica [2016]), and hiring decisions (Reuters [2018]). It is therefore not surprising that as AI usage keeps growing, organizations are under increasing pressure to ensure the fairness of their machine learning

2nd Workshop on Regulatable ML at NeurIPS 2024.

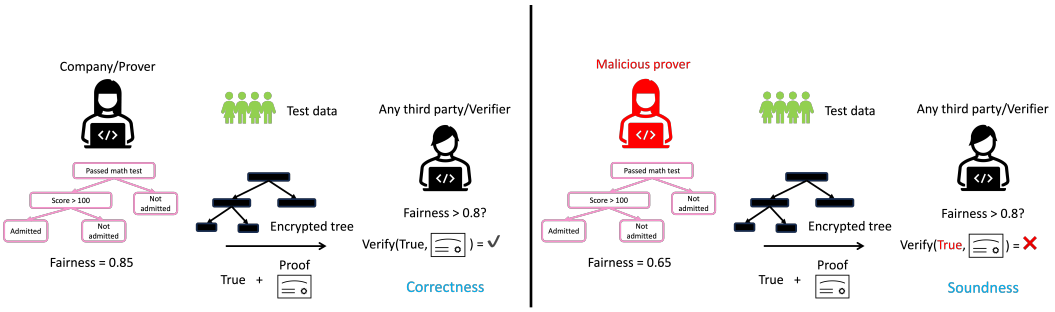

Figure 1: Left: An honest prover computes the fairness outcome and produces the proof. By correctness of the zero-knowledge scheme, verifier accepts. Right: A malicious prover with the model that does not pass the fairness check attempts to cheat during the proof generation. By soundness of the zero-knowledge scheme, verifier rejects.[1]

systems – driven not only by societal expectations (Times [2020]), but also by emerging legal regulations (European Commission [2024], House [2022]). In the U.S., in order to settle a lawsuit filed by the Department of Justice, a well-known tech company has recently built a new system to address algorithmic discrimination in advertising, which was in violation of the Fair Housing Act (Times [2022]). In Europe, the recently introduced AI Act references terms such as "equality", "discrimination", and "discriminatory" forty-eight times, and it emphasizes the importance of implementing non-discrimination measures from the earliest stages of AI model design (European Commission [2024], Equinet [2024]).

Unfortunately, proving that a given model complies with non-discriminatory regulations is not straight-forward, in particular whenever the model is proprietary and must remain confidential. In practice, an external examiner interacts with the company, posing questions about the model's training process and the measures taken to ensure fairness. This process consumes significant resources and offers little satisfaction to either party — the evaluator must rely on the company's honesty, while the company may feel burdened by the lack of clear, objective criteria or the need to disclose proprietary information.

A recent work by Shamsabadi et al. [2023] ("Confidential-PROFITT") offers a solution by utilizing cryptographic techniques in the context of decision trees in order to obtain *confidential* proofs of fair training. It is important to note that 'fair training' in this context does not imply the model is inherently "fair" by societal standards or according to non-discriminatory laws. Rather, it means that a given fairness parameter (related to, e.g., demographic parity), is above a certain threshold. Specifically, Confidential-PROFITT's fair training algorithm works by recursively partitioning the training dataset based on criteria which capture the information gain with respect to not only class, but also sensitive attributes. This allows to link the information gain with respect to the sensitive attribute to the notion of demographic parity. More importantly, it uses zero-knowledge proofs (Goldwasser et al. [1985]), which allow to prove that a statement is true without revealing any information beyond its validity, and applies these to the training algorithm. In particular, zero-knowledge is a cryptographic concept where one person (the prover) can prove to another (the verifier) that they know something, like a password or a solution to a problem, without revealing any information about what that something actually is. It's like proving you know the answer without showing the answer itself. These proofs further provide correctness (a prover proving a correct statement is always able to convince the verifier), and soundness (a malicious prover cannot convince the verifier of a validity of an incorrect statement) guarantees. This solves the dilemma mentioned in the paragraph above: Now, the company can *formally prove* to the external examiner that the training process adheres to fairness principles, all without disclosing proprietary data or methods. Figure 1 illustrates the message flow between the prover and the verifier, where the prover seeks to demonstrate that the fairness property exceeds the 0.8 threshold without disclosing the underlying decision tree to the verifier.

[1]Icons have been retrieved from Flaticon.com

While Confidential-PROFITT is undoubtedly a step forward towards reconciling fairness and privacy concerns in the context of decision trees, it nevertheless leaves a number of questions open. First, while Confidential-PROFITT positions the solution as a *certificate* which proves fairness, their solution is, in fact, an *interactive* and *designated verifier* protocol, which requires a verifier to remain online and to maintain a secret internal state while the prover generates a proof. Therefore, their solution can be verified only by the examiner who participated in this interactive process – a third party (e.g., a examiner from another jurisdiction, or a customer who wishes to verify the model's fairness) would not be convinced by the iteration transcript, and the verification process would need to be done anew. Although there exist standard cryptographic methods of compiling an interactive scheme into a non-interactive one such as Fiat and Shamir [1987], the underlying zero knowledge proof (Franzese et al. [2021]) of Confidential-PROFITT is not amenable to such generic conversion methods. Although Yadav et al. Yadav et al. [2024] recently introduced a non-interactive and publicly verifiable zero knowledge protocol for checking fairness, their method is tailored to neural networks. Moreover, unlike Confidiential-PROFITT, which allows verifier to check group fairness metrics of the proprietary model, the protocol of Yadav et al. [2024] is specifically designed for *individual fairness*. As such, there is currently no explicit non-interactive and publicly verifiable solution for checking group fairness of (private) decision trees in the literature.

Finally, Confidential-PROFITT *assumes* that the test data which is used during the fairness verification check is identical to the one used for the training. The algorithm ensures that this data is kept private, which is crucial to meet the privacy concerns. However, there is nothing preventing a dishonest company from substituting the test data with a *different* dataset during verification, because the functionality it achieves does *not* guarantee that the private model is correctly derived from the test data. At the same time, an artificially crafted dataset (which is not the same as the training dataset) could make the model pass the fairness check — in fact, a dataset which is as simple as not including one minority group (based on the sensitive attribute) allows *any* decision tree to pass the fairness check. Since the dataset is never revealed to the examiner, there is nothing preventing a malicious company from deceiving the examiner!

1.1 Our Contributions

In this work, we address the concern above. Specifically, we provide the first *non-interactive* proof for fair decision trees. In contrast to Confidential-PROFITT, our solution is further *publicly verifiable*, meaning that a certificate produced by our protocol would convince *any party* that a model satisfies a certain fairness constraint with respect to a given test data, even if the party did not participate during the protocol's execution. We further focus on the case where the test data is *not* the same as the training data and need not be kept secret, e.g., the test data can be one of the publicly available fairness datasets such as Census Income (Repository [1996]), Default Credit (Repository [2009]), and American Community Survey Income Ding et al. [2021]. This allows us to circumvent the attacks based on artificially crafted datasets. Finally, our zero-knowledge techniques are different from those used in Confidential-PROFITT, and our evaluation shows that our solution improves upon previous work in terms of communication bandwidth. While our prover time is higher, we believe that the non-interactive and public verifiability features offer greater practical utility, as they enable the creation of a compact, reusable certificate that multiple verifiers can validate asynchronously.

To summarize, we propose the first non-interactive publicly-verifiable proof for fairness of decision trees. Unlike previous works, it does not assume that the training data is the same as the test data, which enables the examiner to check fairness on publicly available datasets and reduces the impact of artificially crafted test datasets. Finally, our construction improves upon prior work in terms of communication bandwidth.

2 Problem Statement, Notation, and Preliminaries

In our scenario we consider two parties: A prover (i.e., company which holds the proprietary decision tree model) and a verifier (i.e., the examiner). The examiner wishes to verify that

the model satisfies a given fairness metric on a publicly available dataset (such as COMPAS, Adult etc). For simplicity, we assume that the (upper-bound on) depth of the decision tree is public, and that our sensitive attribute, along with the classification output, is binary.

2.1 Notation

We use a bracket notation $[n]$ to denote a set of integers $\{1, \ldots, n\}$. We denote the decision tree by $\mathcal{T}$, the height of the tree by h, and the number of attributes by d, respectively. A datapoint $\mathbf{a}$ is represented by a key-value table, i.e., $\mathbf{a} = \{\mathsf{attr}_1 : \mathsf{val}_1, \ldots, \mathsf{attr}_d : \mathsf{val}_d\}$. The sensitive attribute is denoted by $s \in [d]$. We denote a cryptographic hash function by $\mathsf{H}()$. We denote the number of test data points by n.

2.2 Preliminaries

We now provide background on the important components of our construction. First, we discuss the fairness metrics considered in this work. Next, we give a short overview of decision trees – the machine learning model which we focus on in our solution. Finally, we discuss zero-knowledge proofs – a well-known cryptographic primitive which lets the prover prove to the verifier that a given statement holds in a privacy-preserving manner.

Fairness Over the years, numerous fairness metrics have emerged (Heidari et al. [2019]), each grounded in distinct philosophical and societal perspectives. These notions are often conflicting 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. We focus on the following four traditional notions of group-fairness criteria:

- **Demographic Parity** ignores the ground truth and aims to equalize the probability of a positive classifier output in each sensitive group. More formally:
$$\mathrm{DP}(\hat{Y}; s) = \left|\Pr\left[\hat{Y} = 1 \;\middle|\; s = 0\right] - \Pr\left[\hat{Y} = 1 \;\middle|\; s = 1\right]\right|$$
- **Disparate Impact** considers the ratio of the positive prediction rates of the two sensitive groups:
$$\mathrm{DI}(\hat{Y}; s) = \frac{\Pr\left[\hat{Y} = 1 \;\middle|\; s = 0\right]}{\Pr\left[\hat{Y} = 1 \;\middle|\; s = 1\right]}$$
- **Equalized Odds** aims to equalize the false positive and true positive rates in each sensitive group:
$$\mathrm{EOd}(\hat{Y}; s) = \max_{y \in \{0,1\}} \left|\Pr\left[\hat{Y} = 1 \;\middle|\; s = 0, Y = y\right] - \Pr\left[\hat{Y} = 1 \;\middle|\; s = 1, Y = y\right]\right|$$
- **Equality of Opportunity** is a partial form of equalied odds, and considers the disparity only between true positive rates in the sensitive groups:
$$\mathrm{EOp}(\hat{Y}; s) = \left|\Pr\left[\hat{Y} = 1 \;\middle|\; s = 0, Y = 1\right] - \Pr\left[\hat{Y} = 1 \;\middle|\; s = 1, Y = 1\right]\right|$$

Decision trees Decision tree-based solutions are among the most popular machine learning algorithms, particularly known for their effectiveness in classification problems. Their strong performance, combined with high levels of explainability, makes them a popular choice in practice, specifically for fraud detection and automated trading. For simplicity, in the following we focus on binary decision trees for classification problems. The training of a decision tree typically involves recursively splitting the dataset into subsets from the root to the leaves. Each split is determined by a splitting rule that aims to maximize an objective function, such as information gain. The prediction is done by traversing the path from the tree root to the leave, while checking the threshold of the intermediate node and following the corresponding path based on the decision at each step (see Algorithm 1).

Non-Interactive Zero Knowledge Proof A zero-knowledge proof is a well-known cryptographic technique which allows the prover prove to the verifier that a given statement

Algorithm 1: Decision Tree Inference

Input: Decision tree $\mathcal{T}$, input $\mathbf{a}$.
Output: Classification result.

```
1: Let cur := T.root
2: while cur is not a leaf do
3:     if a[cur.attr] < cur.thr then
4:         cur := cur.left.
5:     else
6:         cur := cur.right.
7: return cur.class
```

holds without leaking any information besides the validity of the statement. For example, it is possible to prove that the output of a decision tree $\mathcal{T}$ on a given public input a equals to y *without revealing anything about* $\mathcal{T}$*'s node conditions*. See Appendix B for more intuition. More formally, we consider a non-interactive zero knowledge proof system (henceforth NIZK) for NP relation $\mathcal{R}$ denoted by a tuple $\mathsf{ZK} = (\mathcal{G}, \mathcal{P}, \mathcal{V})$ of three algorithms:

$\underline{\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)}$ is a setup algorithm that samples a public parameter pp, where λ denotes a security parameter.

$\underline{\pi \leftarrow \mathcal{P}(\mathsf{pp}, \mathsf{x}, \mathsf{w})}$ is a prover that outputs a proof π asserting $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$. If $(\mathsf{x}, \mathsf{w}) \notin \mathcal{R}$, $\mathcal{P}$ outputs $\perp$.

$\underline{b \leftarrow \mathcal{V}(\mathsf{pp}, \mathsf{x}, \pi)}$ is a verifier that outputs a decision bit $b \in \{0,1\}^*$.

Here, the common input x to both parties is called *statement* while $\mathcal{P}$'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 $\mathsf{x} = (a, y)$ and $\mathsf{w} = \mathcal{T}$. We assume ZK to satisfy standard security properties, such as *completeness* (i.e., if $\mathcal{P}$ and $\mathcal{V}$ follow the protocol, $\mathcal{V}$ always accepts), *knowledge soundness* (i.e., if $\mathcal{V}$ accepts the proof generated by a cheating prover $\mathcal{P}^*$, then it must be that $\mathcal{P}^*$ owns a valid witness w satisfying given NP relation w.r.t. statement x), and *zero knowledge* (i.e., if the proof generated by $\mathcal{P}$ leaks nothing except that fact that $\exists \mathsf{w}$ such that $(\mathsf{x}, \mathsf{w}) \in \mathcal{R}$). We refer the reader to standard references such as Goldreich [2001] for more formal definitions.

3 Our Zero Knowledge Proof for Decision Tree Fairness

We now design a mechanism which allows to verifiably prove that a proprietary decision tree model satisfies a fairness check on a given public test data without revealing anything about this model besides 1) the upper-bound on the tree height and 2) that it passes the fairness check. To do so, we design a zero-knowledge proof tailored to the fairness check procedure.

3.1 Overview of zkDTF

At a high level, we follow the approach of Zhang et al. [2020], who designed a zero-knowledge proof system for decision tree accuracy (henceforth zkDTA). Our construction, zero-knowledge proof system for decision tree fairness (henceforth zkDTF), consists of the following steps: **(a)** commitment to the model, **(b)** computation of the outcome, **(c)** zero-knowledge fairness proof. We now explain each of these steps.

Commitment to the model As the first step, the prover computes and publishes a cryptographic *commitment* $\mathsf{com}_\mathcal{T}$ to their decision tree model. This commitment can be instantiated with the hash[2] of the following information for each tree node concatenated with a uniformly random string ρ: The attribute attr being checked, the threshold thr, and

[2]Intuitively, a hash function maps an arbitrary binary string to a binary string of a certain length in a way that it is computationally hard to find an input string that matches a given hash value. It is further hard to find a pair of different messages that have the same hash value.

the pointers to the node's left and right child, denoted by left and right. Note that the commitment satisfies two properties: **(i)** It *hides* the committed information – it is hard to guess x given only the hash of x, and **(ii)** It *binds* the committed data – given a hash h, it is computationally hard to come up with two tuples $(\mathcal{T}, \rho)$ and $(\mathcal{T}', \rho')$ such that they both hash to h and $\mathcal{T} \neq \mathcal{T}'$. Note that randomness ρ is crucial for the hiding property and thus must be kept in secret by the prover.

Computation of the outcome Next, the prover computes the prediction for the given public test dataset $\mathcal{D} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, by evaluating a previously committed tree $\mathcal{T}$ on each data point $\mathbf{a}_i$. For each datapoint $\mathbf{a}_i$, the prover also stores the corresponding path path_i from the root to the leaf node. Finally, the prover uses the computed outcomes $\hat{\mathcal{Y}} = \{\hat{\mathbf{y}}_i\}_{i\in[n]}$ (along with the test data and the corresponding ground-truth $\mathcal{Y}$) to obtain the fairness result. We abstract the process of calculating fairness metric by a function $\mathsf{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s)$, 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 2.2. For instance, to calculate Demographic Parity, one may define

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

and FM can be instantiated for other metrics analogously.

Zero-knowledge fairness proof Assuming that the fairness check passes, in the last step, the prover uses a secure NIZK scheme on the previously computed data (predictions, paths, fairness outcome) along with the public data (test dataset $\mathcal{D}$ and commitment $\mathsf{com}_{\mathcal{T}}$) to show the following statement in zero-knowledge: I know a model $\mathcal{T}$ for which all of the following holds:

(1) It is committed under $\mathsf{com}_{\mathcal{T}}$

(2) $\mathcal{T}$'s prediction for $\mathbf{a}_i$ corresponds to path path_i for each $i \in [n]$

(3) path_i's leaf is $\hat{\mathbf{y}}_i$ for each $i \in [n]$

(4) when the agreed upon fairness metric function FM is computed on the tuple $(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}} = \{\hat{\mathbf{y}}\}_{i\in[n]}, s)$, the outcome is below threshold fthr.

Note that our protocol allows for a flexible choice of the NIZK scheme to prove these conditions. In fact, Step (1)-(3) can rely on existing NIZK schemes for batching a large number of predictions w.r.t. a committed decision tree (e.g., Zhang et al. [2020], Campanelli et al. [2024]). Step (4) requires an additional component. However, it can be computed with standard operations supported by the aforementioned NIZK: at most n addition operations, at most 2 multiplication-by-constant operations and at most 4 comparison operations.[3] The comparison operations are already a part of Step (2), and thus we can reuse the same techniques. Further, note that because we use a secure NIZK, the protocol does not leak predictions $\hat{\mathcal{Y}}$ w.r.t test data $\mathcal{D}$; a prover only leaks the fact that a committed tree $\mathcal{T}$ meets the fairness condition w.r.t. a given test data set, sensitive attribute, and fairness threshold.

3.2 Formal Description

We now give the full description of our protocol. First, we define the syntax of ZKP for decision tree fairness (zkDTF). The protocol consists of a tuple of algorithms $\mathsf{zkDTF} = (\mathcal{G}, \mathsf{Commit}, \mathcal{P}, \mathcal{V})$:

$\underline{\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)}$ is a setup algorithm that samples a public parameter pp.

[3]Note that we do not need division operations or multiplication between two private values, which are more expensive than simple linear operations we rely on. For instance, to perform $|\frac{a}{b} - \frac{c}{d}| \leq \mathsf{fthr}$ one can clear the denominator and check the equivalent condition $|ad - bc| \leq bd \cdot \mathsf{fthr}$. As b, d, fthr are public in the case of DP, EOd, EOp, one can indeed check the equivalent condition without division and multiplication between two private values. In a similar manner, one can clear the denominator of DI to avoid division.

Protocol 1: Zero Knowledge Proof for Decision Tree Fairness

Setup $\mathsf{zkDTF}.\mathcal{G}(1^\lambda)$: Run $\mathsf{pp} \leftarrow \mathsf{ZK}.\mathcal{G}(1^\lambda)$. Output pp.

Commit $\mathsf{zkDTF.Commit}(\mathcal{T}, \rho)$: Run $\mathsf{com}_\mathcal{T} = \mathsf{H}(\mathcal{T}, \rho)$. Output $\mathsf{com}_\mathcal{T}$

Prove $\mathsf{zkDTF}.\mathcal{P}(\mathsf{pp}, \mathsf{com}_\mathcal{T}, \mathcal{T}, \rho, \mathcal{D}, \mathcal{Y}, s, \mathsf{FM}, \mathsf{fthr})$:

1: Compute the prediction $\hat{\mathbf{y}}_i = \mathcal{T}(\mathbf{a}_i)$ for each $i \in [n]$. Let $\hat{\mathcal{Y}} = \{\hat{\mathbf{y}}_1, \cdots, \hat{\mathbf{y}}_n\}$.
2: Compute the fairness outcome according to the fairness metric function: $v = \mathsf{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s)$. Abort if $v > \mathsf{fthr}$.
3: Produce a proof π by running $\mathsf{ZK}.\mathcal{P}$ for the following NP relation:

$$\mathcal{R}_{\mathsf{DTF}} = \left\{((\mathsf{com}_\mathcal{T}, \mathcal{D}, \mathcal{Y}, s, \mathsf{fthr}), (\mathcal{T}, \rho)) \ : \ \mathsf{com}_\mathcal{T} = \mathsf{H}(\mathcal{T}, \rho) \wedge \hat{\mathcal{Y}} = \{\mathcal{T}(\mathbf{a}_i)\}_{i=1}^n \wedge \mathsf{FM}(\mathcal{D}, \mathcal{Y}, \hat{\mathcal{Y}}, s) \leq \mathsf{fthr}\right\}$$

Verify $\mathsf{zkDTF}.\mathcal{V}(\mathsf{pp}, \mathsf{com}_\mathcal{T}, \mathcal{D}, \mathcal{Y}, s, \mathsf{FM}, \mathsf{fthr}, \pi)$: Accept π if and only if $\mathsf{ZK}.\mathcal{V}$ for $\mathcal{R}_{\mathsf{DTF}}$ accepts π.

$\mathsf{com}_\mathcal{T} \leftarrow \mathsf{Commit}(\mathcal{T}, \rho)$ is a committing algorithm that outputs a commitment $\mathsf{com}_\mathcal{T}$, taking a trained decision tree $\mathcal{T}$ and randomness ρ as input. We assume that the commitment scheme satisfies standard binding and hiding properties.

$\pi \leftarrow \mathcal{P}(\mathsf{pp}, \mathsf{com}_\mathcal{T}, \mathcal{T}, \rho, \mathcal{D}, \mathcal{Y}, s, \mathsf{FM}, \mathsf{fthr})$ is a DT fairness prover that outputs a proof π asserting that a committed $\mathcal{T}$ satisfies a certain fairness condition w.r.t. test data set $\mathcal{D} = \{\mathbf{a}_1, \cdots, \mathbf{a}_n\}$, ground truth labels $\mathcal{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$, sensitive attribute s, fairness metric function FM, and threshold fthr. If the fairness condition is not satisfied, $\mathcal{P}$ outputs $\perp$.

$b \leftarrow \mathcal{V}(\mathsf{pp}, \mathsf{com}_\mathcal{T}, \mathcal{D}, \mathcal{Y}, s, \mathsf{FM}, \mathsf{fthr}, \pi)$ is a verifier that outputs a decision bit $b \in \{0,1\}^*$. We describe a way to construct a zkDTF protocol using a NIZK scheme ZK as a subroutine in Protocol 1.

4 Evaluation

4.1 Implementation

We implemented our zkDTF protocol by modifying an existing implementation of zero knowledge proof of accuracy (zkDTA) from TAMUCrypto [2021]. While in their protocol the prover's goal is to verifiably prove accuracy of a proprietary model, in our work the goal is prove that the model satisfies a given fairness check. However, both protocols require Steps (1)-(3), which allow us to reuse a large portion of Zhang et al. [2020]. The critical difference between zkDTF and zkDTA lies in Step (4): while zkDTA counts the number of correct predictions and performs an equality check w.r.t a public accuracy value acc there, zkDTF first computes a fairness metric and compares it against a public fairness threshold fthr. Both cases only involve summation of at most n private values. While zkDTA needs n equality check operations, zkDTF requires a few additional multiplication-by-constant and comparison operations for all fairness metrics considered in Section 2.2. However, as Step (2) (i.e., prediction check for all n data points) already requires $O(nh)$ comparison operations in both zkDTA and zkDTF, we observe that a few additional operations in Step (4) only introduce a negligible difference in performance.

As the publicly available implementation of Zhang et al. [2020] relies on the widely deployed Groth16 NIZK protocol Groth [2016], our implementation of zkDTF also invokes it as a backend. The implementation of Groth16 is inherited from the libsnark library Lab [2012] and we instantiated its parameters with the `BN254` pairing-friendly elliptic curve Jancar [2020]. As the most expensive operation of Groth16 consists of Fast Fourier Transform (FFT) and elliptic curve arithmetic, which are parallelizable, we compiled the libsnark library while enabling parallelized execution of these low-level computations.

Table 1: Performance of our zkDTF protocol for checking Equalized Odds. 'DC' denotes the Default Credit data set; 'CI' denotes the Census Income data set. The running time benchmarks are based on our experiments conducted on an Amazon EC2 c7a.12xlarge instance with 96GB RAM (without parallelization). For completeness, we also restate the benchmarks for Confidential-PROFITT Shamsabadi et al. [2023] in the rows marked by 'CP'. These numbers are based on their experiments conducted on Amazon EC2 instances where prover and verifier are connected over a LAN. Note that, unlike our zkDTF, the verifier of Confidential-PROFITT does not receive any data set as input.

	Setup (sec)	Prove (sec)	Verify (sec)	Proof Size (kB)	Publicly Verifiable	Non-Interactive
DC (Ours)	1430	825.6	< 1	0.2	✓	✓
DC (CP)	–	72.2	72.2	109875	×	×
CI (Ours)	1982	1001.5	< 1	0.2	✓	✓
CI (CP)	–	104.7	104.7	148685	×	×

Table 2: Performance of our zkDTF protocol for checking Demographic Parity (DP) and Equalized Odds (EqOd) of decision trees trained on the American Community Survey (ACS) Income data set. The running time benchmarks are based on our experiments conducted on an Amazon EC2 c7a.12xlarge instance with 96GB RAM. We parallelized FFT and elliptic curve operations of the underlying Groth16 NIZK protocol Groth [2016] by exploiting 48 vCPUs.

	Setup Time (sec)	Prove Time (sec)	Verifier Time (sec)	Proof Size (kB)
DP	220	189	< 1	0.2
EqOd	222	190	< 1	0.2

To generate fairness-compatible decision trees (without verifiability/privacy), we follow the approach of Confidential-PROFITT Shamsabadi et al. [2023], which enhances the traditional decision tree training algorithm with an additional fairness-compatibility check. At a high level, in addition to computing an information gain for each split, the algorithm computes an "unfairness gain" using a calculation similar to the standard Gini impurity, but for a given fairness metric (such as demographic parity, equalized odds etc). The algorithm then only utilizes splits for which the unfairness gains are below a certain threshold. As this is not the focus of our paper, for further details we refer the reader to Shamsabadi et al. [2023].

4.2 Performance

For our experiments, we trained decision trees of height $h = 10$ on the following datasets: Default Credit ($(n, d) = (30000, 23)$) Repository [2009], Census Income ($(n, d) = (45222, 14)$) Repository [1996], and American Community Survey Income ($(n, d) = (45222, 10)$) Ding et al. [2021]. See Appendix A for detailed descriptions. We evaluated the performance of our zkDTF. In Table 1 we summarize performance benchmarks and communication bandwidth of zkDTF for Equalized Odds, using Default Credit and Census Income datasets. Table 1 includes comparison with Confidential-PROFITT Shamsabadi et al. [2023]. In Table 2 we also report the performance of our zkDTF executed on the ACS dataset, for checking Demographic Parity and Equalized Odds. For the latter experiment, we enabled OpenMP with 48 threads in order to showcase our approach is parallelization-friendly.

As the underlying Groth16 NIZK protocol Groth [2016] only outputs three group elements on `BN254` elliptic curves, our zkDTF obtains only modest proof sizes. In contrast, the communication bandwidth of Confidential-PROFITT is orders of magnitude larger. Unlike Confidential-PROFITT, however, our zkDTF additionally takes a data set as input, which is typically about a a few megabytes. Although this might seem appear an additional overhead in the communication bandwidth, an amortized overhead is low in practice because the input data set is one of the widely used reference data sets and thus can be reused in order to verify different proof strings generated by different provers. In terms of prover running time, our

current implementation of zkDTF falls short of Confidential-PROFITT. Although this seems to be a drawback, we highlight the fact that our zkDTF enables a model owner to produce a reusable certificate once and for all and then publish it in a public repository, which can be later validated by potentially many verifiers. In contrast, due to its interactive and designated-verifier nature Confidential-PROFITT requires freshly executing an interactive process for every verifier. As the prover time mainly hinges on the complexity of the backend NIZK scheme (Groth [2016]), we leave for future work the improvement of the prover time by employing a more modern, prover-efficient scheme as a backend of our zkDTF. We also remark that our zkDTF requires a setup phase (i.e., invocation of $\mathsf{pp} \leftarrow \mathcal{G}(1^\lambda)$ in Protocol 1). This is a mandatory phase for any NIZK protocols, while most interactive ZK protocols do not require setup. Although this phase takes a significant amount of time due to the heavy setup operations of Groth16 NIZK, we highlight that this is a *one-time* setup phase, that is, once the public parameters pp have been generated for a particular fairness metric, any prover can reuse the same pp to generate fairness proof for the same metric and for arbitrary trees and datasets of bounded sizes.

5 Conclusion and Future Work

We designed a non-interactive and secure solution to prove that a (proprietary and private) decision tree model satisfies a fairness constraint, i.e., a given fairness metric is above a certain threshold, on a dataset which is supplied by the verifier. At its core, our solution is a custom zero-knowledge proof which is tailored to proving fairness of decision trees.

We implemented our solution for the demographic parity and equalized odds fairness metrics. In contrast to the prior work's Confidential-PROFITT protocol, our solution does not assume that the prover uses the correct dataset to generate the proof, which in combination with our protocol's security guarantees ensures that the prover cannot cheat regarding the model's fairness guarantee. Our protocol is further non-interactive and publicly verifiable, i.e., the fairness certificate can be verified by anyone. In terms of performance, our protocol is orders of magnitude cheaper in terms of the verifier's computation time and communication, which albeit comes at the cost of a higher computation time on the prover's side.

Our solution leaves several interesting questions for future work: First, since there is no consensus on a single "general-purpose" fairness metric, it would be interesting to extend our implementation to let the protocol support further metrics. Second, our protocol is tailored to decision trees. Designing efficient protocols to support further machine learning models, e.g., the widely used XGBoost Chen and Guestrin [2016], is an important direction. Finally, our protocol requires the verifier to supply the test data set (or use a publicly available one), and the prover learns this data. It would be interesting to provide a solution in which the verifier's test data remains *private*. General-purpose solutions for this scenario exist, but these are not yet sufficiently efficient. Obtaining a practical scheme will likely require developing new techniques on the cryptographic side.

Acknowledgment

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.

Elisaweta Masserova was supported by the Carnegie Bosch fellowship, a gift from Bosch and NSF Grants No. 1801369 and 2224279.

References

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.

Matteo Campanelli, Antonio Faonio, Dario Fiore, Tianyu Li, and Helger Lipmaa. Lookup arguments: Improvements, extensions and applications to zero-knowledge decision trees. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part II*, volume 14602 of *LNCS*, pages 337–369. Springer, Heidelberg, April 2024. doi: 10.1007/978-3-031-57722-$2_1$1.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. URL `https://doi.org/10.1145/2939672.2939785`.

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/32e54441e6382a7fbacbbbaf3c450059-Abstract.html`.

Equinet. Coding equality in the eu ai act: Equality bodies rising to the challenge, 2024. URL `https://equineteurope.org/conference-equality-proofing-ai-systems-equality-bodies-rising-to-the-challenge/`.

European Commission. Regulation (eu) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (ec) no 300/2008, (eu) no 167/2013, (eu) no 168/2013, (eu) 2018/858, (eu) 2018/1139 and (eu) 2019/2144 and directives 2014/90/eu, (eu) 2016/797 and (eu) 2020/1828 (artificial intelligence act), 2024. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R1689`.

Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987. doi: 10.1007/3-540-47721-$7_1$2.

Nicholas Franzese, Jonathan Katz, Steve Lu, Rafail Ostrovsky, Xiao Wang, and Chenkai Weng. Constant-overhead zero-knowledge for RAM programs. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 178–191. ACM Press, November 2021. doi: 10.1145/3460120.3484800.

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.

Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001. ISBN 0-521-79172-3 (hardback).

Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985. doi: 10.1145/22145.22178.

Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 305–326. Springer, Heidelberg, May 2016. doi: 10.1007/978-3-662-49896-$5_1$1.

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.

The White House. Blueprint for an ai bill of rights: Making automated systems work for the american people, 2022. URL https://www.whitehouse.gov/ostp/ai-bill-of-rights/.

Jan Jancar. Standard curve database: bn254, 2020. URL https://neuromancer.sk/std/bn/bn254.

Markulf Kohlweiss, Mary Maller, Janno Siim, and Mikhail Volkhov. Snarky ceremonies. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 98–127. Springer, Heidelberg, December 2021. doi: 10.1007/978-3-030-92078-4_4.

SCIPR Lab. libsnark: a c++ library for zksnark proofs, 2012. URL https://github.com/scipr-lab/libsnark.

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.

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.

Reuters. Insight - amazon scraps secret ai recruiting tool that showed bias against women, 2018. URL https://www.reuters.com/article/world/insight-amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK0AG/.

Ali Shahin Shamsabadi, Sierra Calanda Wyllie, Nicholas Franzese, Natalie Dullerud, Sébastien Gambs, Nicolas Papernot, Xiao Wang, and Adrian Weller. Confidential-profitt: 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.

TAMUCrypto. Zkdt, 2021. URL https://github.com/TAMUCrypto/ZKDT$_{\text{r}}$elease/tree/master.

The New York Times. Wrongfully accused by an algorithm, 2020. URL https://www.nytimes.com/2020/06/24/technology/facial-recognition-arrest.html.

The New York Times. Meta agrees to alter ad technology in settlement with u.s., 2022. URL https://www.nytimes.com/2022/06/21/technology/meta-ad-targeting-settlement.html.

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.

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.

A Data Sets

For our experiments, we first trained decision trees of height $h = 10$ on the following datasets:

- Default Credit ($(n, d) = (30000, 23)$) 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 assume the sensitive attribute to be the age. The minority group of young individuals under 25 comprises 13% of the dataset.
- Census Income ($(n, d) = (45222, 14)$) 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. Following Shamsabadi et al. [2023], we assume gender (Male, Female) to be the sensitive attribute, with 68% of the dataset's individuals being Males.
- American Community Survey Income ($(n, d) = (45222, 10)$) Ding et al. [2021]: Same as in the Census Income case, the ground truth label indicates whether an 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.

The first two data sets are the largest ones used by Confidential-PROFITT Shamsabadi et al. [2023]. As the Census Income data set is by now outdated, we also conducted the experiment using a replacement dataset suggested in Ding et al. [2021].

B Intuition for Commitments and Zero Knowledge Proofs

Cryptographic Commitment We recall a cryptographic commitment scheme Blum [1981], which is one of the most basic building blocks in cryptography. On a high-level, a commitment can be seen as a digital analogue of "locked box": a message sender first puts a letter in a box locked with some password and sends it to a receiver. As the box is password-protected, the receiver is not able to read the letter until the sender reveals the password later. At the same time, the sender cannot change the content of the letter once the box is handed over to the receiver.

More formally, a commitment algorithm Commit takes a message m and randomness ρ as input, and outputs a commitment string com. The receiver, upon receiving m and ρ, simply checks that $\mathsf{com} = \mathsf{Commit}(m, \rho)$ to verify the opening information (m, ρ) is correct. In practice, Commit can be efficiently instantiated with standardized cryptographic hash functions such as SHA-256, which outputs a 256-bit digest of the input message. The standard security property of commitment includes *hiding* (i.e., by observing com, the receiver learns nothing about the committed message m) and *binding* (i.e., once com is sent to the receiver, the sender cannot open com to two distinct messages m and m'). We refer the reader to standard textbooks such as Goldreich [2001] for more formal treatments.

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. In practice, a setup phase can be conducted by a trusted third party (e.g., government institute) or by some distributed, multi-party computation protocol in order to remove reliance on the trusted third party Kohlweiss et al. [2021].