Fairness in the Eyes of the Data: Certifying Machine-Learning Models

Shahar Segal
Tel-Aviv University
Tel-Aviv, Israel

Yossi Adi
The Hebrew University of Jerusalem
Jerusalem, Israel

Benny Pinkas
Bar-Ilan University
Ramat Gan, Israel

Carsten Baum
Aarhus University
Aarhus, Denmark

Chaya Ganesh
IISc Bangalore
Bangalore, India

Joseph Keshet
Bar-Ilan University
Ramat Gan, Israel

ABSTRACT

We present a framework that allows to certify the fairness degree of a model based on an interactive and privacy-preserving test. The framework verifies any trained model, regardless of its training process and architecture. Thus, it allows us to evaluate any deep learning model on multiple fairness definitions empirically. We tackle two scenarios, where either the test data is privately available only to the tester or is publicly known in advance, even to the model creator. We investigate the soundness of the proposed approach using theoretical analysis and present statistical guarantees for the interactive test. Finally, we provide a cryptographic technique to automate fairness testing and certified inference with only black-box access to the model at hand while hiding the participants' sensitive data.

CCS CONCEPTS

• **Security and privacy → Cryptography**; • **Computing methodologies → Machine learning**.

KEYWORDS

Fairness, Privacy

ACM Reference Format:
Shahar Segal, Yossi Adi, Benny Pinkas, Carsten Baum, Chaya Ganesh, and Joseph Keshet. 2021. Fairness in the Eyes of the Data: Certifying Machine-Learning Models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society (AIES '21), May 19–21, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3461702.3462554

INTRODUCTION

Machine learning systems are increasingly being used to inform and influence decisions about people, leading to algorithmic outcomes that have powerful personal and societal consequences. For instance, decisions such as (i) *is an individual likely to commit another crime?* [2]; or (ii) *is an individual likely to default on a loan?* [36] are made using algorithmic predictions. This can be concerning given the many documented cases of models amplifying bias and discrimination from the training data [6, 8, 23, 31]. To address this formally, a line of recent works considers *fairness* in classification by proposing notions of fairness based on similarity measures and formalizing variants of this notion that provide guarantees against discrimination [10, 16, 20, 22].

One common scenario in which such a discrimination could potentially happen is a setting with a client and a server. The server classifies queries by a client in an automated way using a machine learning model generated by it. On the other hand, the client wants to make sure its queries are treated fairly and its sensitive data is conserved. If the model itself is not a secret, then a client can potentially run tests (such as the ones implied by the references above) on the model to establish its purported fairness without exposing its data. Making a model public, however, is not always in the interest of the server, since it has invested resources such as expertise, data and computation time for the training – and therefore often wants the model to remain proprietary. Moreover, sharing models may in some cases raise security or privacy concerns. It therefore may be deemed appropriate or necessary to outsource any such test to a semi-trusted third party such as a government entity, which would inspect a model and certify its fairness. This raises our first question:

Question 1: Can we design a framework for verifying the fairness of models, giving guarantees to clients while being practically realizable and keeping the model secret from the clients?

Having such a third party relieves the client from testing fairness, but actually just shifts responsibility to someone who might be more qualified to make a judgement about the model. To minimize the necessary trust between the model owner and the third party, such a test would still be restricted to a black-box scenario. In addition, constructing such a test for establishing fairness guarantees can be difficult on its own. Given that the sources and amount of data is limited, it might be that the third party can only use data in the fairness test that the model owner is familiar with. This might enable the model owner to design an unfair model which successfully passes the examination by the third party.

This raises our second question:

Question 2: Can we design a black-box fairness test that gives guarantees even if the test set is (partially) known?

Here, by a black-box test we mean that a test should only query the model M on different inputs but should not make any assumptions about the actual model parameters.

AIES '21, May 19–21, 2021, Virtual Event, USA.

ACM ISBN 978-1-4503-8473-5/21/05…$15.00
https://doi.org/10.1145/3461702.3462554

Our contributions. In this work, we answer both questions affirmatively. We design an architecture for certifying fairness via verification in machine learning models using three (or more) participants, the model owner (or "server") $\mathcal{S}$, the client C and a trusted third party $\mathcal{R}$ (also called "regulator"). Our architecture uses techniques from cryptography to construct secure protocols for

(1) An interactive test between $\mathcal{S}$ and $\mathcal{R}$ allowing to verify with high probability that a model M provided by $\mathcal{S}$ is fair with respect to a set of pre-defined groups. While ensuring that $\mathcal{R}$ does not learn M. This test considers scenarios where $\mathcal{S}$ *is* or *is not* aware of the test data. $\mathcal{R}$ is not involved in the training of M, it only performs certification.
(2) An interactive computation between $\mathcal{S}$ and C which computes a prediction $\hat{y} = M(x)$ from an input x and a model M. The interactive computation neither leaks M to C nor x to $\mathcal{S}$, and yet makes sure that the model that was used in the prediction has been certified by $\mathcal{R}$ beforehand.

Our work provides fairness tests necessary for these protocols that have black-box to the model and uses existing highly efficient cryptographic primitives to implement the tests securely. While we motivate the underlying ideas of these tests on an intuitive level and give formal arguments for their soundness, we also provide experimental evidence that the hypotheses that make our tests possible are viable. Since secure and privacy-preserving computation of models, for both training and inference, is a very active research area (e.g. Guo et al. [15], Juvekar et al. [18], Kumar et al. [25], Mohassel and Zhang [28], Riazi et al. [29], van der Maaten and Hannun [33]), the performance of the current solutions in this field is continuously improving. As our work assumes the existence of secure protocols for inference, and investigates how to add fairness on top of these in a generic way that is independent of the underlying training algorithm, our approach will benefit in practicality from any independent progress that is made in this direction.

Related work. Fairness in algorithms was first investigated by Friedman and Nissenbaum [13]. Since then, further research into data as a source of unfairness in ML decisions has been done [7, 19]. Baluta et al. [4] showed how to verify properties of a DNN (fairness among them). In their work they encode the network into *Conjunctive Normal Forms* and then test if it will likely fulfill certain logical constraints. In comparison, our approach is independent of the concrete model parameters and architecture.

Several statistical measures of unfairness, and fairness criteria are studied by Feldman et al. [12], Zemel et al. [37]. These and subsequent works achieve statistical notions of fairness through post-processing the training data, and/or by enforcing constraints at training time. Our work differs from this line of research in that we want to guarantee fairness which is enforced obliviously of the training process. Dwork et al. [10] shows that statistical notions of fairness are inadequate, while Corbett-Davies et al. [8] established that model calibration does not rule out unfair decisions. These results emphasize that fairness is nuanced, complicated, application-specific, and can depend on legal and social contexts. In this work, we answer the orthogonal question of designing a fairness test for any machine learning model, *given* an accepted fairness definition.

Most relevant prior work to ours is the study by Kilbertus et al. [21]. In this research the authors suggest to use cryptographic primatives for *fairness certification*, *fair model training*, and *model decision verification*. However, this study was mainly focused on certifying a model via fair training. Additionally it did not provide analysis and guarantees for model fairness certification. Our study focuses on certifying fairness via verification of any existing machine learning models, regardless of the training process. Since the framework is oblivious to the training process, multiple fairness definitions can be certified post-training, even if the training process did not take them into account. We analyze our framework from both theoretical and practical points of view while providing guarantees based on the number of samples available in the test set. We also explore a different scenario where these samples are known to $\mathcal{S}$ during model training, which makes the certification harder.

PRELIMINARIES

Let $\mathcal{X}$ be the set of possible inputs, $\mathcal{G}$ be a finite set of groups that are relevant for fairness (e.g., ethnic groups) and $\mathcal{Y}$ be a finite set of labels. We suppose $\mathcal{X} \times \mathcal{G} \times \mathcal{Y}$ is drawn from a probability space Ω with an unknown distribution $\mathcal{D}$. Let M be a trained model for a classification task of $\mathcal{D}$, we denote $M(x)$ for classification of input $x \in \mathcal{X}$.

The goal of training a model M is usually to achieve low error on unseen data. In addition, when dealing with model fairness, we also take into account a measurement with respect to $\mathcal{G}$. While there are plenty of fairness measurements [35], here we focus on group risk and likelihood based definitions, specifically: *overall risk equality* (ORE), *equalized odds* (EO) and *demographic parity* (DP). First, we define the conditional *risk* and *likelihood* respectively:

$$\ell_g(M) = \mathop{\mathbb{E}}_{(x,g',y')\sim\mathcal{D}} \left[\mathbb{I}\left\{M(x) \neq y'\right\} | g' = g\right] \quad \text{(Risk)}$$

$$\ell_{g,y}(M) = \mathop{\mathbb{E}}_{(x,g',y')\sim\mathcal{D}} \left[\mathbb{I}\left\{M(x) \neq y'\right\} | g' = g, y' = y\right] \quad \text{(Risk with label condition)}$$

$$L_{g,y}(M) = \mathop{\mathbb{E}}_{(x,g',y')\sim\mathcal{D}} \left[\mathbb{I}\left\{M(x) = y\right\} | g' = g\right] \quad \text{(Likelihood)}$$

where $\mathbb{I}\{\pi\}$ is an indicator function with a predicate π. The empirical conditional risk is defined for a given independent sample set $T = \{(x_1, g_1, y_1), ..., (x_m, g_m y_m)\} \sim \mathcal{D}^m$ as:

$$\bar{\ell}_g(M, T) = \frac{1}{m_g} \sum_{i=1}^{m} \mathbb{I}\{M(x_i) \neq y_i \wedge g_i = g\} \quad (1)$$

where m_g is the number of samples in T from group g.

We define a metric called the *fairness gap* to be the maximal margin between any two groups (and labels). Formally, we use three well-known measurements:

$$\max_{g_0,g_1\in\mathcal{G}} |\ell_{g_0}(M) - \ell_{g_1}(M)| \quad \text{(ORE)}$$

$$\max_{g_0,g_1\in\mathcal{G}, y\in\mathcal{Y}} |\ell_{g_0,y}(M) - \ell_{g_1,y}(M)| \quad \text{(EO)}$$

$$\max_{g_0,g_1\in\mathcal{G}, y\in\mathcal{Y}} |L_{g_0,y}(M) - L_{g_1,y}(M)| \quad \text{(DP)}$$

Likewise, the *empirical fairness gap* (EFG) is defined using the empirical approximation of each measurement respectively.

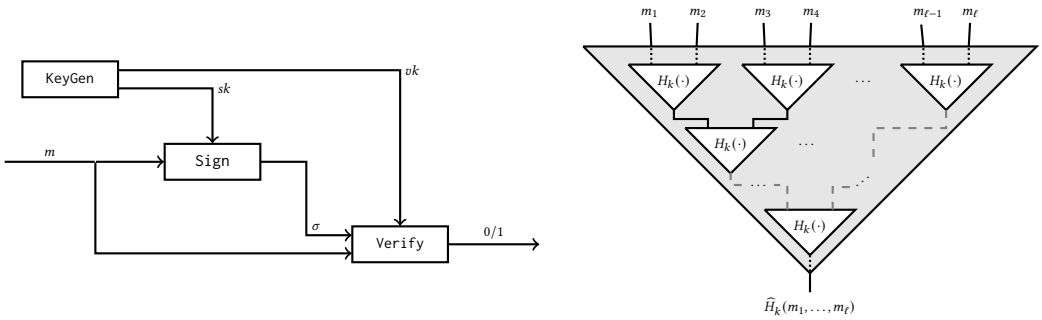

Figure 1: Signatures (left), Merkle Trees (right)

Lastly, we call model M ϵ-**fair** on $(\mathcal{G}, \mathcal{D})$ with respect to a fairness measurement, if its fairness gap is smaller than ϵ with confidence $1-\delta$, which is similar to PAC-style fairness [30]. A model M is then called ϵ-**fair** on $(\mathcal{G}, \mathcal{D})$ under the ORE metric if:

$$\Pr\left[\max_{g_0,g_1\in\mathcal{G}} |\ell_{g_0}(M) - \ell_{g_1}(M)| > \epsilon\right] \leq \delta \tag{2}$$

Plugging-in the fairness gap metric for EO and DP yields the corresponding ϵ-fairness definitions.

CRYPTOGRAPHIC PRIMITIVES

We now describe the cryptographic primitives that are necessary to implement the proposed framework in more detail: *Signatures*, *Collision-Resistant Hash Functions* and *Secure Computation*.

Signatures. Cryptographic signatures can be thought of as a computational analogue to hand-written signatures. We give a schematic explanation of signature schemes in Figure 1. Here, a pair of a public verification key vk and a secret signing key sk are generated together by the key generation algorithm `KeyGen`. sk will be used by the signing algorithm `Sign` to create a signature σ on a message m, while the verification algorithm `Verify` decides if a pair (m, σ) is valid according to the verification key vk or not. Secure signature schemes guarantee *unforgeability*, which means that given vk and arbitrarily many signature pairs $\{(m_i, \sigma_i)\}_{i\in[\ell]}$, it is hard to generate a valid signature σ on a message m, where $m \neq m_i$ for all $i \in [\ell]$.

Collision-Resistant Hashing. We will use a *Collision-Resistant Hash Function* $H_k : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^n$, which is an efficiently computable function such that it is hard for any polynomial-time algorithm (in n) that is given a random k to come up with x_1, x_2 such that $H_k(x_1) = H_k(x_2)$. In practice, one uses e.g. SHA-3 to implement H_k for a k that is fixed in advance. Since the input length of SHA-3 is fixed, in order to hash longer messages, one can apply H_k recursively using a Merkle Tree (see Figure 1). For such a Merkle Tree it can be proven that if H_k is collision-resistant then $\hat{H}_k$ is too.

Secure Computation. We further let parties perform computations on shared data such that the computation does not reveal their inputs, for purposes as mentioned in the next Section.

Secure Computation can be imagined as the existence of a "trusted third party" $\mathcal{F}_{\mathsf{SC}}$ which performs a computational task for certain parties. $\mathcal{F}_{\mathsf{SC}}$ would receive the inputs from both participants, do the computation, and send the output to the participants. The task of this party is outlined in Figure 2. As is common in the secure computation literature, this description assumes that the computation is done by a circuit K. Participant $\mathcal{P}_1$ provides to the trusted party its input x_1, while participant $\mathcal{P}_2$ provides its input x_2. The trusted party computes $K(x_1, x_2)$ and sends its outputs to the respective participants. By this definition this "idealized box" $\mathcal{F}_{\mathsf{SC}}$ achieves the desired privacy objective.

Two parties $\mathcal{P}_1, \mathcal{P}_2$ can talk to this trusted third party.

Input: Upon message $(\texttt{Input}-\mathcal{P}_1, x_1)$ from $\mathcal{P}_1$ and $(\texttt{Input}-\mathcal{P}_2, x_2)$ from $\mathcal{P}_2$ store x_1, x_2 locally.

Compute: Upon input $(\texttt{Compute}, K)$ from $\mathcal{P}_1$ and $\mathcal{P}_2$ and if x_1, x_2 have been stored:

(1) Check if x_1, x_2 have suitable size for the circuit K. If not, output `(Abort)`.

(2) If x_1, x_2 have suitable size then compute $(y_1, y_2) = K(x_1, x_2)$ and store y_1, y_2 locally.

Output: Upon input `(Output)` from $\mathcal{P}_1$ and $\mathcal{P}_2$ and if y_1, y_2 have been computed, send y_1 to $\mathcal{P}_1$ and y_2 to $\mathcal{P}_2$.

Figure 2: A Trusted Third Party $\mathcal{F}_{\mathsf{SC}}$ for Secure Computation.

Such a trusted third party $\mathcal{F}_{\mathsf{SC}}$ as described in Figure 2 does not necessarily exist in the real world, but *it can be emulated* using cryptographic tools as a protocol consisting of two (or more) entities sending messages to each other over a network. Guarantees in these protocols can be given if at least one of the participants is acting honestly throughout the process.

The two most popular approaches for implementing Figure 2 are based on cryptographic paradigms called *Fully Homomorphic Encryption* (FHE) and *Secure Multiparty Computation* (MPC). For comparison, current FHE schemes are constrained by their demand for computational power and they at best can evaluate a few hundred AND-gates of the circuit K per second. MPC on the other hand, which has a higher demand in terms of communication, can achieve a much better throughput. In particular, there exist MPC schemes that are tailored at efficiently implementing the function $M(\cdot)$ [1] [5, 9].

[1] A recent framework for privacy preserving machine learning using MPC built on PyTorch: CrypTen [11].

THE FRAMEWORK

We present our framework from a broad overview, and leave most of the implementation details to later sections. There we describe two interactive tests to verify fairness and more wholesome view on the cryptographic aspects. For now we focus on the general flow and interaction between the different participants we previously described. We also make their roles more explicit and describe the security guarantees that are given to each of them as well as the trust relations. Note that we discuss our framework with respect to three participants but it can easily be generalized to any larger number. In particular, it allows for a large number of regulators $\{\mathcal{R}_i\}_{i=1}^{k}$ that a client C can choose from or even perform the regulators role by itself.

- The *Server* $\mathcal{S}$ initially generates the model M. Its main objective is to keep M secret. He may try to use an unfair model towards $\mathcal{R}$ or C.
- The *Client* C has a private input x and wishes to obtain $\hat{y} \leftarrow M(x)$, where the server provides M. The objective of C is to ensure that M is fair while keeping x private.
- The third participant is the *Regulator* $\mathcal{R}$ who should neither learn M nor x or y. After $\mathcal{S}$ proves the fairness of model M, $\mathcal{R}$ outputs certificate $\mathtt{cert}_M$ for the model to attest its validity. $\mathtt{cert}_M$ is tied to another certificate $\mathcal{R}$ issued, $\mathtt{cert}_{ID}$, which serves as the identity of $\mathcal{R}$. When $\mathtt{cert}_M$ is shown to C it can verify that indeed $\mathcal{R}$ certified the model using $\mathtt{cert}_{ID}$. In addition, $\mathcal{R}$ has access to the sample set T in order to check fairness, which is possibly known to $\mathcal{S}$.

In terms of modeling security we assume that $\mathcal{S}$ will try by any means to get an unfair model certified, use an unfair model to a client or learn C's input. In particular, $\mathcal{S}$ may actively deviate from any specified program or protocol in any way. On the other side, we consider that C and $\mathcal{R}$ would follow the protocol but may try to learn information about $\mathcal{S}$'s model in the process. In more cryptographic terms we consider a malicious $\mathcal{S}$ while C and $\mathcal{R}$ are semi-honest. We further assume that either party is computationally polynomially time-bounded.

The certificates $\mathtt{cert}_{ID}, \mathtt{cert}_M$ are implemented using a digital signature scheme and a collision-resistant hash function. Roughly, $\mathtt{cert}_{ID}$ is a public verification key that is tied to the identity of $\mathcal{R}$, and $\mathtt{cert}_M$ is a signature on a compressed version of the model that is computed using a collision-resistant hash function. Here, a cryptographic signature ensures that only $\mathcal{R}$ could issue $\mathtt{cert}_M$, while the hash function forces $\mathcal{S}$ to use the same model with C that he used when obtaining $\mathtt{cert}_M$.

At the beginning of the protocol, $\mathcal{R}$ generates its public certificate $\mathtt{cert}_{ID}$ and makes it available. $\mathcal{S}$ and $\mathcal{R}$ then interact to generate the model certificate $\mathtt{cert}_M$ for model M. In the process $\mathcal{R}$ is allowed to query M an arbitrary number of times to ensure fairness. To perform an inference by $\mathcal{S}$ and C, both first agree on a regulator certificate $\mathtt{cert}_{ID}$ that they will use. Then, C obtains an output $\hat{y}$ based on its input x and on a model M' provided[2] by $\mathcal{S}$. Here, C only accepts $\hat{y}$ if M' is certified by the regulator behind $\mathtt{cert}_{ID}$

[2]Here we write M' to denote that technically a malicious $\mathcal{S}$ could try to perform the inference with whatever model M' it wants. The job of the inference algorithm is then to enforce that $M' = M$ for some previously certified M.

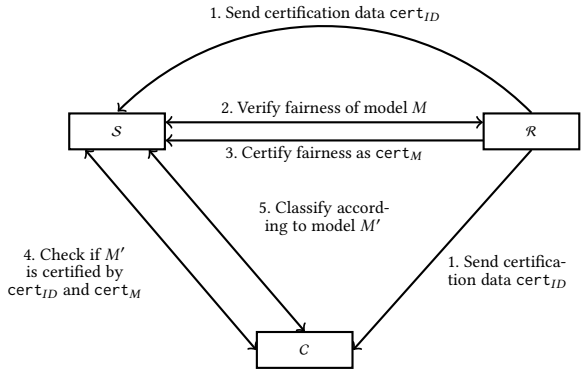

Figure 3: Certification and Verification

for fairness. C does not learn anything about M', besides $\hat{y}$. Fig. 3 describes the aforementioned process schematically.

Both the inference on M and the verification of $\mathtt{cert}_M$ that are necessary in Fig. 3 could be done easily if $\mathcal{S}, \mathcal{R}$ and C would have access to a "trusted third party" $\mathcal{F}_{\mathsf{SC}}$ which performs the computational task for them. That trusted third party would receive inputs from participants, do the computation, and send the output back to them. In our case, $\mathcal{F}_{\mathsf{SC}}$ would receive all secret input from the participants and send $\mathtt{cert}_M$ to $\mathcal{S}$ after verifying M is fair. $\mathcal{F}_{\mathsf{SC}}$ can also send $M(x)$ to C if model M' is certified in that manner. As mentioned in the previous section, such a "trusted third party" can be emulated using cryptographic protocols for Secure Computation. The use of cryptography guarantees that the protocol is secure against a *malicious* $\mathcal{S}$ or semi-honest $C, \mathcal{R}$.

VERIFYING FAIRNESS INTERACTIVELY

We now turn to introduce two interactive tests which allow $\mathcal{R}$ to determine if a model M is ϵ-fair:

(1) The model M is queried using a sample set T which is *unknown* to $\mathcal{S}$. We show that fairness guarantees about M can be made given by the empirical fairness gap (EFG) and a lower-bound on the minimal size of the set T with respect to each group $g \in \mathcal{G}$.
(2) The model M is queried using a sample set $\tilde{T}$ which is derived from a set T in an augmented and randomized fashion. The set T as well as the augmentation algorithm are known to $\mathcal{S}$ in advance. We show that this test implies ϵ-fairness of M, given that the augmentation impacts fair and unfair models in a different way.

Both of the aforementioned tests are independent of the representation of M, make no requirement on its training algorithm and only require access to $M(\cdot)$ for different inputs. All claims below are based on ϵ-fairness with confidence $1 - \delta$ under the ORE fairness metric, however these can be modified to other group-based fairness definitions.

Verifying Fairness using Private Data. The simpler case is when the model M is queried using an i.i.d sample set T which is *unknown* to $\mathcal{S}$. This setup is rather standard in machine learning, as verification using i.i.d samples can be achieved via classic concentration bounds from PAC learning [32, 34]. We apply similar bounds to

assess the minimum number of samples needed for fairness verification.

Denote the *Empirical Fairness Gap* as

$$EFG = \max_{g_0,g_1 \in \mathcal{G}} |\bar{\ell}_{g_0}(M,T) - \bar{\ell}_{g_1}(M,T)|.$$

The following states the conditions which guarantee that a model is ϵ-fair with a confidence $1-\delta$.

Claim 1. *A model M is ϵ-fair with confidence $1-\delta$ if:*

$$EFG < \epsilon \text{ and } \min_{g \in \mathcal{G}} m_g \geq \frac{2}{(\epsilon - EFG)^2} \ln \frac{2|\mathcal{G}||\mathcal{Y}|}{\delta} \tag{3}$$

for $T = \{(x_1, g_1, y_1), ..., (x_m, g_m, y_m)\} \sim \mathcal{D}^m$, where m_g, as in Eq. (1), denotes the number of occurrences of g in T.

DP and EO can be achieved by using the corresponding EFG definition and minimizing over $\mathcal{G} \times \mathcal{Y}$, counting m_g and $m_{g,y}$ respectively in (3). The full proof of the above claim can be found in the supplementary materials.

Verifying Fairness using Augmented Data. The disadvantage of the aforementioned test is that all test data T must be hidden so that the model generator $\mathcal{S}$ cannot use it to adapt M accordingly. In other settings, we would like to test for fairness using public data, which can be known to $\mathcal{S}$. This setting is realistic in many scenarios. For example, if labelled data is costly, getting unique labelled data for a test will be difficult for $\mathcal{R}$.

A straightforward argument against this approach is once the data is publicly available, T is not chosen independently of M. Thus, a malicious $\mathcal{S}$ can create an unfair model that memorizes the set T and responds fairly on it, so that it passes the test outlined above. To counter such dishonest training, we need a method to alter the existing samples and force some sort of generalization abilities. We therefore define the notion of an *augmentor*. An augmentor applies random augmentations to the input which alter the sample but still preserves its label and group with high probability. Here we use it to generate *new* samples for querying that with high probability were not seen during model training. This is a necessary but not sufficient condition in order to ensure a valid test for M. For example, consider an augmentor that only alters the first few pixels of the image. A model that simply ignores those pixels can still overfit on the rest of the image and pass any test.

Hence, we suggest to use a set of randomized augmentation functions to reduce memorization capabilities of an adversary. For this, the assumption is that ϵ-fair models behave differently from unfair models when queried against the samples augmented by the augmentor. Then, this different behavior can be leveraged to expose the unfair nature of certain models. Our approach follows this assumption to construct a querying test set in the same fashion of the test that was previously defined.

More specifically, define an algorithm augmentor $\mathsf{aug} : \mathcal{X} \times \mathcal{G} \times \{0,1\}^\tau \to \mathcal{X}$ that gets as input a random string and a sample and outputs a new augmented sample. The label and group of the new sample should be the same as the original sample with high probability.

We re-define the conditional risk to be on an augmented sample from $\mathcal{D}$. Formally:

$$\ell_{g,\mathsf{aug}}(M) = \mathop{\mathbb{E}}_{(x,g,y)\sim\mathcal{D}, r\leftarrow\{0,1\}^\tau} \Big[\mathbb{I}\{M(\mathsf{aug}(x,g;r)) \neq y\} \Big].$$

As mentioned before, there is no guarantee that samples augmented by aug yield better results than T itself. We need an additional assumption on the behavior of fair and unfair models when shown augmented samples from aug, thus we call a class of models $\mathcal{M}$ *detectable* if it fulfills that assumption.

Definition 1 (($\epsilon, \alpha, \mathsf{aug}$)−detectable fairness)**.** Let $\mathcal{A}$ be an arbitrary training algorithm which outputs a model in $\mathcal{M}$. $\mathcal{M}$ has $(\epsilon, \alpha, \mathsf{aug})$-detectable fairness on $\mathcal{D}$ if there exists $m \in \mathbb{N}$ such that for any $T \sim \mathcal{D}^m$ and $M \sim \mathcal{A}(\mathcal{D}, T, \mathsf{aug}, \alpha)$, M is ϵ-fair if:

$$\max_{g_0,g_1 \in \mathcal{G}} |\ell_{g_0,\mathsf{aug}}(M) - \ell_{g_1,\mathsf{aug}}(M)| \leq \alpha \ .$$

Definition 1 allows us to build an interactive test and to empirically find parameters ϵ, α, m and an augmentor for which it appears to be true. The intuition behind Definition 1 is that in order to cheat the test, M is required to behave differently on the augmented data than on the new unobserved samples. In practice, augmentations are commonly used to improve robustness and generalization, thus M is less likely to be able to generalize on a large set of them without the risk of exposing its unfair nature, as aug challenges its generalization capabilities. The parameterization yields a non-trivial angle both for breaking our overall construction and for improving it.

Notice, the above definition does not imply that all ϵ-fair models have this property, and some fair models will not be discovered due to that. Empirically, we observed that such models can efficiently be detected and we demonstrate that in the experimental section. Additionally, we observe that the output of aug is not required to be indistinguishable from a new sample from $\mathcal{D}$. In particular the definition does not rule out that $\mathcal{A}$ is aware of the possible augmentations.

Let $\tilde{T}$ be a sample set T after augmenting each sample. Denote $\bar{\ell}_{g,\mathsf{aug}}(M, \tilde{T})$ the empirical conditional risk and

$$EFG = \max_{g_0,g_1 \in \mathcal{G}} \left| \bar{\ell}_{g_0,\mathsf{aug}}(M, \tilde{T}) - \bar{\ell}_{g_1,\mathsf{aug}}(M, \tilde{T}) \right|.$$

We state the following,

Claim 2. *Let $\mathcal{M}$ be a class of models with $(\epsilon, \alpha, \mathsf{aug})$-detectable fairness. Let $T, \tilde{T}$, aug and $M \in \mathcal{M}$ be as stated above. M is ϵ-fair with confidence $1-\delta$ if:*

$$EFG < \alpha \text{ and } \min_{g \in \mathcal{G}} m_g \geq \frac{2}{(\alpha - EFG)^2} \ln \frac{2|\mathcal{G}||\mathcal{Y}|}{\delta} \ .$$

In other words, we can certify ϵ-fairness of a model with high confidence assuming $(\epsilon, \alpha, \mathsf{aug})$-detectable fairness. The proof is in the supplementary materials.

IMPLEMENTING THE FRAMEWORK

We describe how to implement the framework from previous sections using the interactive tests and guarantees from Claim 1. While the implementation is described at a high level, it is easy to instantiate each of the components based on existing cryptographic tools and the experimental results in the Experiments section.

Creating a Verification Test

Consider a design of an interactive test based on the set $T = \{(x_1, g_1, y_1), ..., (x_m, g_m, y_m)$ and parameters δ, ϵ as follows:

(1) The regulator $\mathcal{R}$ computes the minimal m_g fulfilling Eq. (3) by assuming $EFG = 0$. If T does not contain enough samples from each group, then $\mathcal{R}$ aborts. If $\mathcal{R}$ does not abort, it tells $\mathcal{S}$ the total number of inputs m that will be checked.
(2) $\mathcal{R}$ and $\mathcal{S}$ run a secure computation of a functionality $\mathcal{F}_{\mathsf{Check}}$ which is described below. $\mathcal{S}$ inputs M into $\mathcal{F}_{\mathsf{Check}}$ while $\mathcal{R}$ inputs $(\{(x_i, g_i, y_i)\}_{i\in[m]})$. The functionality $\mathcal{F}_{\mathsf{Check}}$ consists of the following steps:
   (a) Compute $\hat{y}_i \leftarrow M(x_i)$ for all $i \in [m]$.
   (b) For all $i \in [m]$, compute a bit b_i as 1 if $\hat{y}_i = y_i$ and 0 otherwise.
   (c) Compute for each group g the empirical risk $\bar{\ell}_g(M, T)$ based on Eq. (1) and the b_i values.
   (d) Evaluate Eq. (3) of Claim 1 (checking ϵ-fairness). Output 1 if the statement holds and 0 otherwise.

Based on the statement of Claim 1 it follows that $\mathcal{F}_{\mathsf{Check}}$ will output 1 if and only if the model M provided by $\mathcal{S}$ is ϵ-fair with confidence $1 - \delta$.

The secure computation of $\mathcal{F}_{\mathsf{Check}}$ implements the functionality as a Binary circuit K that is evaluated on secret inputs. We examine the size of this circuit in a following section under Efficiency.

A test using augmented data. If $\mathcal{R}$ instead wishes to use public and augmented data as for Theorem 2 then this will only work assuming that M is (ϵ, α)-detectable as defined in Definition 1. In such a setting $\mathcal{R}$ would now create a test set T' from T locally using an augmentor aug and then follow the exact same path as for the public data (albeit with different constants).

Algorithms

We now describe how to use the circuit K from the previous section to implement the framework. The overall approach is as follows: Initially, $\mathcal{R}$ generates a signature key pair and distributes the verification key to all other participants. Then $\mathcal{R}$ and $\mathcal{S}$ run a secure computation which runs the interactive test and computes a Merkle tree hash $\hat{H}_k(M)$ of the model M. If the test finds that the model is fair, then $\mathcal{R}$ signs ($\hat{H}_k(M), \epsilon, \delta$, fairness definition string) and sends the signature to $\mathcal{S}$. By signing ϵ, δ and a fairness definition string we allow multiple fairness definitions and hyperparameters to be certified.

Later, whenever $\mathcal{S}$ and C run a certified inference for ϵ, δ and a fairness definition, then in addition to running a secure computation of $M(x)$, the functionality will also recompute the hash $\hat{H}_k(\cdot)$ of the model provided by $\mathcal{S}$ and output it to C, while $\mathcal{S}$ sends the signature on the model to C. C can then locally check if $\mathcal{R}$ originally issued the signature on the hash for those hyperparameters and fairness definition, given the public verification key of $\mathcal{R}$. The overall protocols are outlined in Figure 4.

Security

We provide a sketch of the argument about the security of $\pi_{\mathsf{Framework}}$. This must naturally stay on a high level, since we did not make the security properties of the framework formal.

We will have three participants $\mathcal{S}, C, \mathcal{R}$ as outlined before. Let $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme and $H_k(\cdot)$ be a collision-resistant hash function whose key k is a common input to all parties. Moreover, let $\mathcal{F}_{\mathsf{SC}}$ be a functionality for secure computation as outlined in Figure 2. $\mathcal{S}$ has a model M as input, $\mathcal{R}$ has a fairness validation set T as well as parameters ϵ, δ and $fair$ the fairness definition string.

Setup: This reflects Step 1 of Figure 3.
(1) $\mathcal{R}$ uses KeyGen to generate a key pair (sk, vk). $\mathcal{R}$ keeps sk private and sends vk to $C, \mathcal{S}$ as cert_{ID}.

Certification: This reflects Steps 2, 3 of Figure 3.
(1) $\mathcal{S}, \mathcal{R}$ input the same values into $\mathcal{F}_{\mathsf{Check}}$.
(2) Let K be the circuit as outlined in the previous section. Create a circuit K_{cert} that performs the following: First run K on the respective inputs as before, computing $\mathcal{F}_{\mathsf{Check}}$. Denote the output bit of this circuit as b. Then compute the Merkle tree output $h \leftarrow \hat{H}_k(M)$ based on the hash function H_k. Finally, output (b, h) to $\mathcal{R}$.
(3) Both parties run a secure computation of K_{cert} using $\mathcal{F}_{\mathsf{SC}}$.
(4) If the output bit b is 1, then $\mathcal{R}$ computes $\sigma(h) \leftarrow \mathsf{Sign}_{sk}(h, (\epsilon, \delta, fair))$ and sends it as cert_M to $\mathcal{S}$.

Inference: This reflects Steps 4, 5 of Figure 3.
(1) $\mathcal{S}$ sends σ to C. C also knows the signature verification key vk.
(2) $\mathcal{S}$ and C run a secure computation, where $\mathcal{S}$ inputs $\tilde{M}$ and C inputs its input x. For this secure computation they construct a circuit K_{inf} as follows:
   (a) Compute $\hat{y} \leftarrow \tilde{M}(x)$.
   (b) Compute $\tilde{h} \leftarrow \hat{H}_k(\tilde{M})$.
(3) $C, \mathcal{S}$ run a secure computation of K_{inf} using $\mathcal{F}_{\mathsf{SC}}$. C obtains as output $(\hat{y}, \tilde{h})$ while $\mathcal{S}$ does not obtain anything.
(4) C computes $b \leftarrow \mathsf{Verify}_{vk}(\tilde{\sigma}, (\tilde{h}, \epsilon, \delta, fair))$. If this is true then C accepts $\hat{y}$. Otherwise it rejects it.

Figure 4: Protocol $\pi_{\mathsf{Framework}}$ for Certified Inference

First, we note that $\pi_{\mathsf{Framework}}$ leaks to $\mathcal{R}$ and C the Merkle-tree hash h of the model. But since it can be assumed that M has high entropy and the implementation of H_k is a cryptographic hash function, the leakage of h should be tolerable.[3]

That being said, we base our security argument on statements about the security of the building blocks that are used, which can be instantiated using well-known cryptographic constructions:

- *The functionality $\mathcal{F}_{\mathsf{SC}}$ can be implemented using a secure protocol.* As mentioned above this can be done using secure two-party or multi-party computation (MPC).
- *There exist secure signature and hashing schemes.*

Given these primitives, we can assume that the certification and inference steps of Figure 4 are as secure as if they were computed by a trusted party: Assume that in the inference step, the signature $\tilde{\sigma}$ and output $\tilde{h}$ of $\mathcal{F}_{\mathsf{SC}}$ are validated. This can only happen due to 3 cases: (i) $\tilde{\sigma}$ was generated for $\tilde{M}$ by $\mathcal{R}$ (which is the desired course of events); (ii) $\tilde{\sigma}$ was issued by $\mathcal{R}$ but for a different $\tilde{M}'$; or (iii) $\tilde{\sigma}$

[3] It is possible in principle to reduce this leakage by computing $\mathcal{R}$'s signature of h, and the signature verification by C, in a secure computation, but this will considerably increase the overhead. In the other direction, if we are willing to leak some more information then the circuit K can be modified to output to $\mathcal{R}$ whether M successfully classified each input x_i and let $\mathcal{R}$ compute the ϵ-fairness of the model locally. This will simplify the secure computation at the cost of leaking more data to $\mathcal{R}$.

was never issued by $\mathcal{R}$. In the last case, $\mathcal{S}$ must have broken the security of the signature scheme. In the second case, $\mathcal{S}$ must have broken the collision-resistance of H_k. Therefore, either $\mathcal{S}$ managed to break the signature scheme or the hash function, or $\mathcal{R}$ signed $\tilde{M}$. $\mathcal{R}$ computes this signature if and only if the model passed the interactive test. Based on the statement of Theorem 1 it follows that this test passes if and only if the model $\tilde{M}$ provided by $\mathcal{S}$ is ϵ-fair with confidence $1-\delta$.

Efficiency

We now estimate the efficiency of implementing our framework using $\pi_{\texttt{Framework}}$. We first claim that it only makes sense to run our framework in settings where the ML inference is done using a secure computation: If the inference is not computed using a secure computation, then one option is for the client to learn the model and run by itself a check for fairness, or send the model to another party and ask it to do this check. Another option is that the client simply hands over its input to the model owner, but this would require prohibitively expensive zero-knowledge proofs, to be computed at the owner side, to attest to fairness of the output without revealing anything about the model.

Therefore, given that inference is done via secure computation, the parties must incur the cost of running a secure computation of the inference, and the efficiency of the framework should be measured by the additional overhead that is added on top of the secure inference.

The main computational tasks that are run by $\pi_{\texttt{Framework}}$ are as follows:

- The **Certification** phase runs m instances of a secure computation of inference and in addition computes a hash of the model and checks the accuracy of the output.
- The **Inference** phase runs a single secure computation of the inference and in addition computes a hash of the model.

The **Certification** phase is a one-time event, and therefore its overhead is less critical. Theorem 1 shows that the number of samples m_g per group should be $m = \frac{2}{(EFG-\epsilon)^2} \ln \frac{2|G|}{\delta^2}$. Setting for example $EFG = 0.05, \epsilon = 0.1, \delta = 0.2$ and considering $|G| = 100$ groups, we get that $m_g \approx 6800$, which does not seem to be too far off from existing training set sizes.

In more detail, we describe here the cost of implementing the different steps of the circuit K which computes the certification: Step (a) needs to implement the inference m times. This is by far the largest component of the circuit. Step (b) computes m comparisons, which are easy. Step (c) computes $\bar{\ell}_g(M,T)$ for each group g, based on Eq. 1. This computation must sum the b values for each group g. To make this step efficient, the circuit must hardwire the connections for these summations, and the locations of the inputs from each g can be known. (There is no need to hide these locations from $\mathcal{S}$.) Eq. 1 also computes a division by m_g, but there is no need to compute the division and the circuit forwards $m_g \cdot \bar{\ell}_g(M,T)$ to the next step. Step (d) tests Eq. 3 for each pair of g_0, g_1, namely computes $\bar{\ell}_{g_0}(M,T) - \bar{\ell}_{g_1}(M,T)$. Since the input to this step is $m_{g_i} \cdot \bar{\ell}_{g_i}(M,T)$ then the test in this equation should be changed appropriately (which is straightforward, especially if $m_{g_0} = m_{g_1}$).

As for the cost of computing $M(\cdot)$, current secure computation implementations for this task only hide the weights of a DNN but reveal the actual network structure and activation functions. We assume that our secure computation will also only hide the weights as this seems to be a standard assumption. Therefore, we ask what is the additional cost of hashing this data over the default cost of using the weights in the computation of the model.

There is a lot of current work on lightweight hashing schemes for usage in zero-knowledge proofs, and it is reasonable to expect that a lot of improvements in this area will be made in the near future. As a baseline, we consider the Keccak-F function, which is the basis of the SHA3 standard. That function takes a 1600 bit input and can be implemented by a Boolean circuit of 38,400 AND gates (see Abril et al. [1]), i.e. 24 AND gates per input bit. If we use a Merkle tree then the total number of hashes is twice the number of input blocks[4]. Therefore, the total cost is about 48 AND gates per input bit.

Now, with regards to the secure evaluation of the model (not considering special MPC implementations for secure inference[5]), let us consider a setting where the weights have 32 bit fixed-point values. The cost per each weight (when used in DNN inference) must be at least that of multiplying the weight with either an input or output of a hidden layer and adding all these products together (neglecting the cost of the activation function). Multiplying the weight with a 32 bit value costs 185 ANDs per input bit, while adding up the result would only require 6 ANDs per input bit (see [1]), and we therefore take the assumption that the total cost of the secure computation is 191 AND gates per bit of the weights (neglecting the activation function). Therefore the fairness verification increases the cost of inference in this model by only about 25%. While using optimized implementations for inference will make the additional overhead from hashing larger, we can in practice lower the cost of hashing drastically by exploiting special properties of $\mathcal{F}_{\mathsf{SC}}$ which allow the use of homomorphic commitments. We leave such specialized hashing techniques as interesting future work.

EXPERIMENTS

We provide empirical evidence to demonstrate that the assumptions made for the fairness tests are meaningful.

We used six different datasets from various domains: visual (UTK-Faces [38], LFW [17], Colored-MNIST [3], and a subset of CelebA[6] [27]), tabular (Adult Income [24]) and spoken (TIMIT [14]). The datasets vary in size and disparity of minority groups and as such some can be used to create fair or unfair models based on their empirical fairness gap (EFG). We demonstrate the variety of our datasets and detail the preprocess in supplementary materials.

[4]We can improve on that by having the circuit output to C the results of the first layer of the Merkle tree, and have C locally compute the rest of the tree. For this to work, we will on the other hand have to add random values to each input block to avoid lookup table-based attacks on preimages of H_k.

[5]This analysis neglects recent works such as e.g. [5] that apply to special types of networks only. We believe that the accuracy of the networks such as MobileNets that are used in [5] is too low to be of use for fairness testing.

[6]We annotated 8,500 celebrities out of 10,177 in the dataset for ethnicity using Amazon Mechanical Turk. Three turkers annotated three images of each of the 8,500 celebrities, resulting in 177,683 images. The annotations can be downloaded from www.github.com/will/be/published/.

Table 1: Fairness test in the private and public settings on the 3 image datasets. "Regular" refers to the model trained fairly, while "Bias" refers to the biased sampled training.

Dataset	Model	Private Setting				Public Setting			
		Accuracy		ORE EFG		Accuracy		ORE EFG	
		Regular	Bias	Regular	Bias	Regular	Bias	Regular	Bias
UTKFace	ResNet18	89.76	88.56	0.012	0.093	96.11	91.44	0.027	0.139
C-MNIST	LeNet	98.11	74.01	0.001	0.450	89.17	67.97	0.007	0.340
CelebA	ResNet18	97.63	96.95	0.007	0.034	96.60	97.02	0.010	0.033

Private Data Setup. In the following setup, we assume that $\mathcal{R}$ possesses a subset of secret samples to be used to certify a model M for fairness and accuracy. Naturally, we split the data into a training and test subset. Setting ϵ-fair and δ-confidence thresholds, we can certify whether a model is fair using the conditions in *Claim* 1. A bottleneck of these conditions is our dependency on the size of the sample set. Datasets with bigger sample set allow us to certify more (fair) models, while we were not able to certify a (fair) model if the sample set was too small, even if it is indeed truly fair under the chosen fairness metric.

We performed our test on the mentioned datasets with $\delta = 0.05$ and $\epsilon \in \{0.05, 0.075, 0.1\}$. For some tasks this gap and confidence level might be intolerable, but for others, such as gender prediction of a face image, which is the task set for UTKFace, LFW and CelebA, it is better than the existing empirical gaps between ethnicity groups of well-known service providers' models [6].

The test results for ORE are shown in Figure 5. As shown, out of the six datasets only C-MNIST and CelebA produced fair models during our training for $\epsilon = 0.05$, while UTKFace has a fair model for $\epsilon = 0.075$. LFW, Adult Income and TIMIT datasets are all below the threshold of all tests, either due to sample size or a large EFG. Therefore, we focus on the first three datasets as they are the only ones to pass any of our tests. Note that by adjusting the allowed bias, ϵ, we can certify the other datasets. For example, the minority group in LFW has only 559 samples. With its current empirical gap, $EFG = 0.049$, choosing $\epsilon = 0.2$ would suffice to certify the LFW model.

To further evaluate the setup, we trained the same models with a tainted batch sampler. The sampler showed less samples from the smallest minority group-label pair (g_1, y_1) in each batch in order to generate a synthetic sample disparity. We denoted these models as *Bias* in Fig. 5 and Table 1. The taint resulted in an almost as accurate model with a much larger EFG, suggesting they are less fair. For EO, only CelebA had enough samples to certify a model for $\epsilon = 0.05$. It requires at least twice as many samples (since we count $m_{g,y}$ instead of m_g). We find it interesting as it implies the amount of data should be a consideration even for which definition of fairness is practical to choose. We detail the results for EO and DP metrics in the supplementary materials. Results for bias C-MNIST does not appear in Figure 5 since its performance is significantly worse.

Augmented Public Data Setup. In this setup, all the data used in the test is known to all participants. With that in mind, we show a potential augmentor for datasets of images and demonstrate empirically that unfair models cannot pass our test as both accurate and fair. Even though the training set is the same as the test set, the key difference is that $\mathcal{S}$ fixes M before we apply our augmentor to the dataset with new randomness. This generates diverse enough samples, for which models that are fair and generalize well on augmented samples pass the test, while models which are either unfair or bad at generalization fail the test. Our augmentations include rotation, cropping, blanked pixels [39] and added Gaussian noise. Each augmentation was set to be invoked at a certain probability threshold which was chosen randomly. The augmented images should keep the same label and group as the original image to the human eye. By doing so, we hope to generate varied data that cannot be easily reversed or overfitted on.

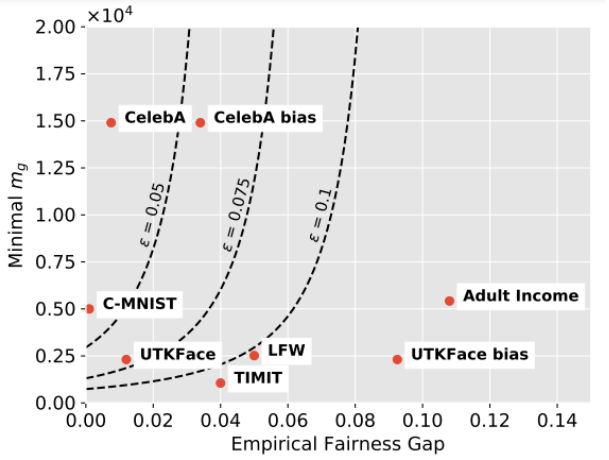

Figure 5: Private fairness test borders by EFG and the minimal m_g. Left to the dashed border is the area where a model would pass the test for that ϵ with $\delta = 0.05$. Dots indicate the ORE results for each dataset.

We tested our three image datasets using the ORE metric, the results are in Table 1. We used the same method to generate fair and unfair models as in the private data section, except that during training we used the augmentor per sample to generate a new augmented sample each time. When we trained the models on the original dataset, the models were not able to generalize on the augmented data.

The results show that there exists a margin in EFG between the fair and unfair models on UTKFace, C-MNIST and CelebA, while the margin is different between datasets, potentially due to their varying size and different complexities of the tasks. This suggests the existence of some α per dataset, based on Definition 1, but

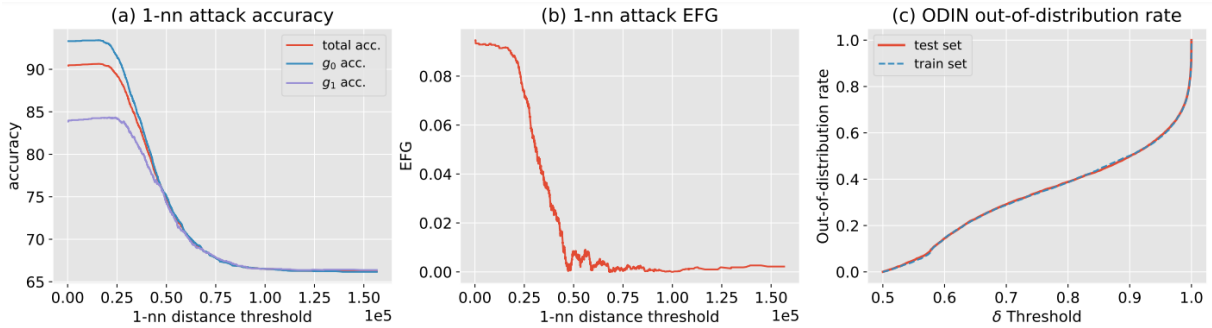

Figure 6: (a)-(b) 1-NN attack accuracy and EFG; (c) ODIN out-of-distribution rate.

we were not able to pinpoint the exact α. We conducted further attempts to characterize α in supplementary materials.

Attacks Against Public Fairness Tests. As we assume that our test works without knowledge of the concrete model, our scheme might be susceptible to an indirect attack on our augmentor. For example, if the model could distinguish between the public data available and a new sample, it could try and behave fairly during the test, but unfairly when an actual new sample is shown. To mimic such an attack, we tested on UTKFace whether it is easy to fool our test using a simple k-nearest neighbour algorithm (kNN) or an out-of-distribution detection technique , ODIN [26], on top of a fair classifier to identify the augmented samples. For a fixed threshold distance from our augmented dataset, we switch to the unfair model and otherwise output the class identified by the kNN. For the ODIN attack, we create a threshold to detect out-of-distribution samples to switch to the unfair classifier. Ideally these attacks use a fair classifier to pass the test as fair when needed, while future new samples (being "far enough" in threshold terms) invoke the unfair model as predictor. We gave the kNN augmented samples of the test as referenced neighbors and plotted the accuracy and EFG by the threshold distance for $k = 1$ in Fig. 6a-b (larger k had worse results). In order to have a similar EFG to the fair model, it has to suffer a drop in accuracy; from 91.44% to 81.9%.

In the other attack, we tuned ODIN's hyperparameters, taking the values which had at least 95% success rate identifying test samples and had the best results at detecting new samples as out-of-distribution. Further details on tuning are listed in the supplementary materials. We plotted the train and test detection rate as out-of-distribution by threshold in Fig. 6c. As can be seen, the sets are detected at similar rates, and are nearly indistinguishable. This resulted in a similar fair or unfair behavior depending on the chosen threshold.

These experiments suggest that these types of attacks are not a good approach to attack the proposed verification test, as this hybrid models cannot pass as both fair and accurate enough for practical applications.

DISCUSSION & FUTURE WORK

We present an interactive test to verify fairness of any machine learning model using cryptographic tools. The interactive test ensures $\mathcal{R}$ does not learn M, x does not leak to $\mathcal{S}$, and M does not leak to C, yet it verifies the model was used during inference has been certified by $\mathcal{R}$. We experimented with two scenarios where the test data is either public or private. We provide analysis and guarantees for the test data, as well as rigorously define the relation between the empirical fairness gap to the sample set sizes.

Moreover, from our guarantees and experiments we noticed not all fairness definitions are created equally, some are harder to verify and require a much larger volumes of data, i.e. EO requires at least twice as many samples as ORE. This leaves room for consideration on what practical definition should we aim for with respect to limited resources or what compromise needs to be made in terms of fairness gap and certainty (ϵ and δ).

For future work we would like to further explore the public data scenario. Specifically, to characterize the detectable fairness hyper-parameter α and its relation to other parameters like the sample set size T, the amount of randomness used per augmentation, etc. Additionally, we would like to explore whether these parameters can be estimated *in advance*, without having to conduct experiments on a dataset. Results suggest that this is a challenge on its own. Moreover, as we are dealing with large models we also require to hash the model inside secure computation. This step has substantial cost, and it is an open question if it could be made more efficient in practice using different ideas than ours. Lastly, the proposed method is focused on group-based fairness definitions, exploring other fairness definitions is also an interesting research direction.

REFERENCES

[1] Victor Arribas Abril, Pieter Maene, Nele Mertens, and Nigel Smart. [n.d.]. 'Bristol Fashion' MPC Circuits. https://homes.esat.kuleuven.be/~nsmart/MPC/. Last accessed on 11/16/2019.
[2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine Bias: there's software used across the country to predict future criminals. And it's biased against blacks. ProPublica 2016.
[3] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. 2019. Invariant risk minimization. *arXiv preprint arXiv:1907.02893* (2019).
[4] Teodora Baluta, Shiqi Shen, Shweta Shinde, Kuldeep S Meel, and Prateek Saxena. 2019. Quantitative verification of neural networks and its security applications. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 1249–1264.
[5] Assi Barak, Daniel Escudero, Anders Dalskov, and Marcel Keller. 2019. Secure Evaluation of Quantized Neural Networks. Cryptology ePrint Archive, Report 2019/131. https://eprint.iacr.org/2019/131.
[6] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. 77–91.
[7] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science* 356,

6334 (2017), 183–186.
[8] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 797–806.
[9] Ivan Damgård, Daniel Escudero, Tore Frederiksen, Marcel Keller, Peter Scholl, and Nikolaj Volgushev. 2019. New primitives for actively-secure MPC over rings with applications to private machine learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1102–1120.
[10] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*. ACM, 214–226.
[11] FacebookResearch. 2019. CrypTen. https://github.com/facebookresearch/CrypTen.
[12] Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 259–268.
[13] Batya Friedman and Helen Nissenbaum. 1996. Bias in computer systems. *ACM Transactions on Information Systems (TOIS)* 14, 3 (1996), 330–347.
[14] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. 1993. DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon technical report n* 93 (1993).
[15] Chuan Guo, Awni Hannun, Brian Knott, Laurens van der Maaten, Mark Tygert, and Ruiyu Zhu. 2020. Secure multiparty computations in floating-point arithmetic. *arXiv preprint arXiv:2001.03192* (2020).
[16] Úrsula Hébert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. 2018. Multicalibration: Calibration for the (computationally-identifiable) masses. In *International Conference on Machine Learning*. 1944–1953.
[17] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Technical Report 07-49. University of Massachusetts, Amherst.
[18] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 1651–1669.
[19] Matthew Kay, Cynthia Matuszek, and Sean A Munson. 2015. Unequal representation and gender stereotypes in image search results for occupations. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM, 3819–3828.
[20] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2018. Preventing Fairness Gerrymandering: Auditing and Learning for Subgroup Fairness. In *International Conference on Machine Learning*. 2569–2577.
[21] Niki Kilbertus, Adrià Gascón, Matt J. Kusner, Michael Veale, Krishna P. Gummadi, and Adrian Weller. 2018. Blind Justice: Fairness with Encrypted Sensitive Attributes. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 2635–2644.
[22] Michael P Kim, Aleksandra Korolova, Guy N Rothblum, and Gal Yona. 2019. Preference-Informed Fairness. *arXiv preprint arXiv:1904.01793* (2019).
[23] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2017. Inherent Trade-Offs in the Fair Determination of Risk Scores. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.
[24] Ron Kohavi. 1996. Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. to appear.
[25] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2019. Cryptflow: Secure tensorflow inference. *arXiv preprint arXiv:1909.07814* (2019).
[26] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690* (2017).
[27] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.
[28] Payman Mohassel and Yupeng Zhang. 2017. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 19–38.
[29] M Sadegh Riazi, Christian Weinert, Oleksandr Tkachenko, Ebrahim M Songhori, Thomas Schneider, and Farinaz Koushanfar. 2018. Chameleon: A hybrid secure computation framework for machine learning applications. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. 707–721.
[30] Guy N Rothblum and Gal Yona. 2018. Probably approximately metric-fair learning. *arXiv preprint arXiv:1803.03242* (2018).
[31] Rachael Tatman and Conner Kasten. 2017. Effects of Talker Dialect, Gender & Race on Accuracy of Bing Speech and YouTube Automatic Captions.. In *INTERSPEECH*. 934–938.
[32] Leslie G Valiant. 1984. A theory of the learnable. *Commun. ACM* 27, 11 (1984), 1134–1142.
[33] Laurens van der Maaten and Awni Hannun. 2020. The Trade-Offs of Private Prediction. *arXiv preprint arXiv:2007.05089* (2020).
[34] Vladimir Vapnik. 2006. *Estimation of dependences based on empirical data*. Springer Science & Business Media.
[35] Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *2018 IEEE/ACM International Workshop on Software Fairness (FairWare)*. IEEE, 1–7.
[36] Kaveh Waddell. 2016. How algorithms can bring down minorities' credit scores. *The Atlantic* 2 (2016).
[37] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. 2013. Learning fair representations. In *International Conference on Machine Learning*. 325–333.
[38] Zhifei Zhang, Yang Song, and Hairong Qi. 2017. Age Progression/Regression by Conditional Adversarial Autoencoder. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.
[39] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2017. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896* (2017).