## Appendix A. Implementation of DP

Keeping in mind the dangers of implementing DP with floating point arithmetic (**?**), we stick with the best practice of using fixed-point and integer arithmetic as recommended by, for example, OpenDP [6]. We implement our DP mechanism using their discrete representations. We use 32 bit precision to ensure correctness and did not observe any loss in accuracy due to this when compared with accuracy in-the-clear.

## Appendix B. Description of MPC Schemes

A variety of MPC schemes are available in the literature that can be employed based on different security threat models (see Tab. 3). These are defined based on either the number of computing parties that can be corrupted or the modified behavior of the corrupted parties. MPC schemes for the *dishonest-majority* setting offer security even if half or more number of computing parties are corrupted, while MPC schemes for the *honest-majority* setting offer security as long as more than half of the computing parties remain uncorrupted. If the corrupted computing parties adhere to the MPC protocols but try to gather additional information, they are said to be corrupted by a *passive* adversary, whereas if the corrupted computing parties deviate from the protocol instructions, they are said to be corrupted by a *active* adversary. We design our MPC protocols in Sec. 4 building on the primitive MPC protocols from the schemes in Tab. 3. Our protocols are sufficiently generic to be used in dishonest-majority as well as honest-majority settings, with passive or active adversaries. This is achieved by changing the underlying MPC scheme to align with the desired security setting.

The MPC schemes in Tab. 3 above provide a mechanism for the servers to perform cryptographic primitives through the use of secret shares, namely addition of a constant, multiplication by a constant, and addition of secret-shared values (which can all be done by the servers through carrying out local computations on their own shares) and multiplication of secret-shared values (which requires communication among the servers and is denoted as MPC protocol $\pi_{\mathsf{MUL}}$ in this paper). Building on these cryptographic primitives, MPC protocols for other operations have been developed. We use a secure division protocol $\pi_{\mathsf{DIV}}$ with an iterative algorithm that is well known in the MPC literature (Catrina and De Hoogh, 2010)). We also use a secure comparison protocol $\pi_{\mathsf{GTE}}$: at the start of $\pi_{\mathsf{GTE}}$, the parties have secret sharings $[\![x]\!]$ and $[\![y]\!]$ of integers $x$ and $y$; at the end of the protocol they have a secret sharing of 1 if $x \geq y$, and a secret sharing of 0 otherwise (Keller, 2020). The protocol $\pi_{\mathsf{EQ}}$ for equality test works similarly.

## Appendix C. Description of Protocol $\pi_{\mathsf{CF}}$

For protocol $\pi_{\mathsf{CF}}$, the MPC servers receive as input secret-shares of a vector $Y$ with ground truth labels, secret-shares of a vector $\hat{Y}$ with predicted labels, and secret-shares of a vector $S$ with sensitive attribute values. All vectors have length $N$, which is the total number of instances. The MPC servers execute protocol $\pi_{\mathsf{CF}}$ to compute as output secret-shares of TP, TN, FP, and FN for each group, i.e. the protected group $p$ and the unprotected group $s$. We use the logic in the truth table shown in Table 4 to compute the values of TP, FN, FP, and TN and replace control flow statements (if-then-else) by multiplications, as commonly

6. https://opendp.org/

done in MPC to make the number and kind of computations done during protocol execution independent of specific values of the data. Multiplying the columns in Table 4 with the value $S[i]$ of the sensitive attribute as in the equations below, gives the contributions of instance $i$ to the TP, FN, FP, and TN scores of the protected and unprotected groups:

We expand the above 8 equations and rewrite them as in Table 5 by precomputing the common products

$$\text{tp} = Y[i] \cdot \hat{Y}[i], \quad \text{ys} = Y[i] \cdot S[i], \quad \text{ps} = \hat{Y}[i] \cdot S[i], \quad \text{tps} = \text{tp} \cdot S[i]$$

for each instance to reduce the number of multiplications.

$$\begin{aligned}
\text{TP(p)} &= \sum_{i=1}^{N} Y[i] \cdot \hat{Y}[i] \cdot S[i] & \text{TP(u)} &= \sum_{i=1}^{N} Y[i] \cdot \hat{Y}[i] \cdot (1 - S[i]) \\
\text{FP(p)} &= \sum_{i=1}^{N} (1 - Y[i]) \cdot \hat{Y}[i] \cdot S[i] & \text{FP(u)} &= \sum_{i=1}^{N} (1 - Y[i]) \cdot \hat{Y}[i] \cdot (1 - S[i]) \\
\text{FN(p)} &= \sum_{i=1}^{N} Y[i] \cdot (1 - \hat{Y}[i]) \cdot S[i] & \text{FN(u)} &= \sum_{i=1}^{N} Y[i] \cdot (1 - \hat{Y}[i]) \cdot (1 - S[i]) \\
\text{TN(p)} &= \sum_{i=1}^{N} (1 - Y[i]) \cdot (1 - \hat{Y}[i]) \cdot S[i] & \text{TN(u)} &= \sum_{i=1}^{N} (1 - Y[i]) \cdot (1 - \hat{Y}[i]) \cdot (1 - S[i])
\end{aligned}$$

where $S[i] \in \{u, p\}$ is mapped to $S[i] \in \{0, 1\}$ for mathematical computations ($u = 0$ and $p = 1$).

Following the above logic, the parties, on Lines 4–17 of protocol $\pi_{\mathsf{CF}}$, compute the contribution of an instance towards TP, FN, FP, or TN for the protected or unprotected group. On Lines 5–8, the parties precompute the secret shares of the common products (tp, ys, ps and tps) for each instance to reduce the number of multiplications.

Table 4: Logic for evaluating TP, FN, FP, and TN

| $Y[i]$ | $\hat{Y}[i]$ | true pos. $Y[i] \cdot \hat{Y}[i]$ | false neg. $Y[i] \cdot (1 - \hat{Y}[i])$ | false pos. $(1 - Y[i]) \cdot \hat{Y}[i]$ | true neg. $(1 - Y[i]) \cdot (1 - \hat{Y}[i])$ |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |

On Lines 9–16, the parties compute the secret shares of TP, FN, FP, or TN based on the formulas in Table 5 for the protected and unprotected groups.

## Appendix D. Training Details

We evaluated `PrivFairFL` on two datasets, details of which are given below. [7]

**ADS.** The ADS dataset[8] is a collection of 300 advertisements explicitly rated by 120 users (Roffo and Vinciarelli, 2016). Additionally, the dataset contains demographic information,

7. The implementations are all single-threaded.
8. https://www.kaggle.com/groffo/ads16-dataset

Table 5: Formulas for computing TP, FN, FP, and TN for protected and unprotected groups per instance

| | Protected group | Unprotected group |
|---|---|---|
| TP | tps | tp − tps |
| FP | ps − tps | $\hat{Y}[i]$ − ps − tp + tps |
| FN | ys − tps | $Y[i]$ − tp − ys + tps |
| TN | $S[i]$ − ys − ps + tps | 1 − $S[i]$ − $Y[i]$ + ys − $\hat{Y}[i]$ + ps + tp − tps |

interests, personality traits and textual phrases describing the likes and dislikes of each user. The advertisements are in the form of pictures. Each user rated each advertisement on a 1–5 scale. Following the categorization in (Roffo and Vinciarelli, 2016), we labeled any ratings above 3 as "positive" and below and equal to 3 as "negative".

We extracted features from the advertisements (number of faces, label types, safe category, objects) using Google Vision API.[9] The features for the users are based on one-hot-encoded information on their gender, country, interests, and income category, the first two indices of the zipcode, and their normalized age, along with the word embeddings[10] for the words describing the likes and dislikes of users. Each training sample consists of the features of a user, features for an ad and the explicit rating provided by the user for the particular ad, leading to 300 samples for each user. We excluded all the users who had only negative ratings resulting in a total of 109 users and a total of 32700 training samples. We performed a stratified split over these users, keeping 80% of their data as training data and the remaining 20% data as the test data.

To predict if a user is interested in an ad ("positive" instance) we trained a dense layer with 2 units each with sigmoid activation. In the centralized setup (CL), this model is trained for 630 epochs with a batch size of 64. For the federated setup (FL), the models are trained for 800 rounds with each client training the local models for 2 epochs per round with a batch size of 24. For the models that we train with DP-SGD, the micro-batches are set same as the batch size with clipping threshold as 1 and noise multiplier as 0.7. We use SGD as the optimizer with glorot initializer and learning rate of 0.1, and binary cross-entropy as loss to train all the models. All the above models are trained thrice with different values of seeds (47568, 42, 1000), and the average values of the metrics are reported.

**ML-1K.** The Movielens dataset contains explicit 5-star movie ratings from 6040 users on 3952 movies. All users have rated at least 20 movies, but the count varies per user. Movie and user features are available, including demographic user information. The classification task is to predict whether a user would enjoy a particular movie. For this, we convert greater than 3 star reviews to a positive rating, and 3 stars or less to a negative rating, as with the ADS dataset. For movie features, we include one-hot-encoded genre and time between the user rating and the movie release year. For user features, we include categorical age, gender, and one-hot-encoded occupation.

We take a sub-sample of 75 users for the experiments, within which the percentage of positive ratings is similar to the whole dataset (sub-sample: 58.11%, whole dataset: 57.52%),

9. https://cloud.google.com/vision/docs/object-localizer
10. Based on pretrained Glove embeddings.

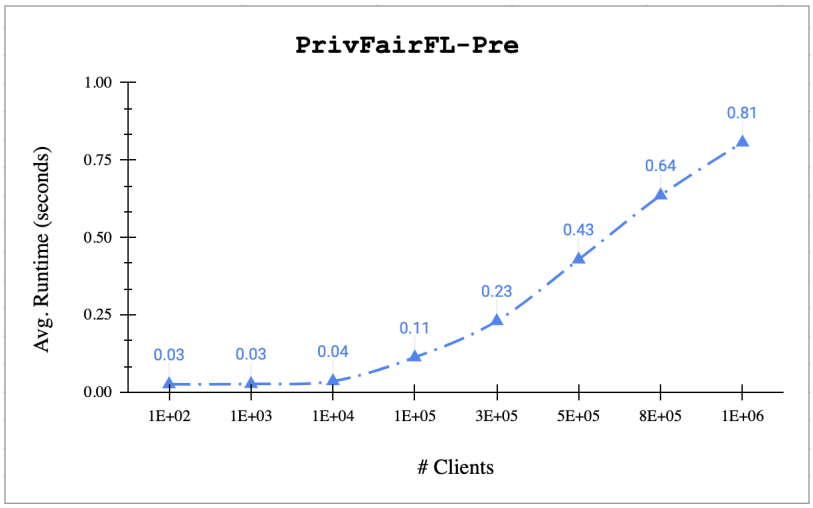


Figure 1: Runtimes for 3PC passive security setting for `PrivFairFL-Pre`

as is the percentage of female users (sub-sample: 32%, whole dataset: 28.29%). The model consists of a dense layer with 1 unit and sigmoid activation. In the centralized setting (CL), the model is trained for 20 epochs with a batch size of 32 and learning rate 0.015. In the federated setting (FL), the model is trained with 2 local epochs per user, 300 rounds of federated learning, with a learning rate of 0.03. The batch size and micro-batch size for DP-SGD are set to the size of the users' training set. The clipping threshold is set to 1.0 and the noise multiplier to 1.1. The model is trained with an SGD optimizer and binary cross-entropy loss. All results are reported on averages from 3 runs, using random seeds 1, 2 and 3.

**Training with `PrivFairFL`.** We recall that our proposed techniques are independent of the model being trained. The model training times are thus the same as for training any DP-models in FL, and the cost for privacy-preserving debiasing with `PrivFairFL` is separate. Our experiments show that it takes about 32-40 minutes to train a LR model with DP-SGD on 109 clients with 800 federated runs. We also note that `PrivFairFL-Post` is independent of any hyperparamters, giving the same convergence guarantees as state-of-the-art FL+DP training techniques. `PrivFairFL-Pre` might benefit from additional rounds of FL or local epochs of the clients depending on the dataset. The number of rounds can be set by the aggregator in FL, who keeps track of the loss and utility on the held-out validation set and has knowledge on the kind of data. Additionally, we propose to use early stopping per client and setting a large maximum number of epochs. The convergence guarantees of `PrivFairFL-Pre` thus also follow from the convergence guarantees of FL+DP.

## Appendix E. Additional Experiments

Our contributions lie in achieving group fairness using simple yet effective bias mitigation techniques used in practice – for imbalanced datasets and for biased predictions – while preserving the privacy of the users. We ran experiments to evaluate the scalability of our proposed protocols and to study privacy-fairness-accuracy trade-offs.

**Scalability of proposed MPC protocols.** We provide complete and formal privacy guarantees by heavily relying on MPC protocols at the expense of longer runtimes. We note

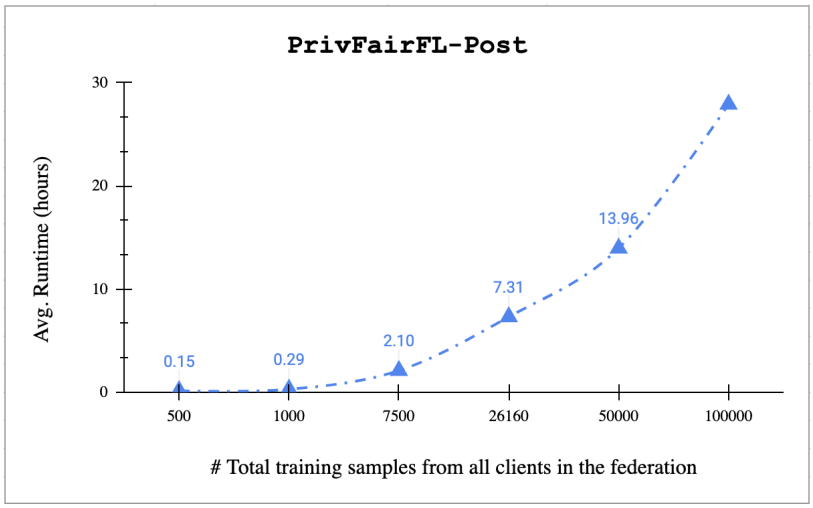


Figure 2: Runtimes for 3PC passive security setting for `PrivFairFL-Post`

that our proposed techniques are independent of the training phase of the model and that the runtimes of the MPC protocols for bias mitigation are a one-time reasonable price to achieve both privacy and fairness.

`PrivFairFL-Pre` essentially collects the statistics from each client and aggregates them to publish DP-weights for further federated learning. This is equivalent to collecting one vector from each client (one datapoint), making the aggregation for bias mitigation dependent only on the number of clients in the federation. Figure 1 shows the runtimes to execute $\pi_{\mathsf{RW}}$ with the number of clients in the federation ranging from 100 to 1,000,000, demonstrating that the runtimes increase linearly with the number of clients (note that the scale on the horizontal axis is logarithmic).[11] We see that for hundreds of thousands of clients, it takes about 0.8 seconds to debias the distributed dataset. While these runtimes may only roughly approximate what one could expect during actual deployment in dynamic environments, they indicate that our MPC protocol for preprocessing is practical enough to achieve both privacy and fairness.

`PrivFairFL-Post` collects the training labels, the inferred labels for the training data with the (unfair) model, and the sensitive attributes from each client in the federation to compute DP ROC curves over all the training data from all the clients, where each client shares multiple datapoints. This makes our postprocessing technique dependent on the total number of training samples rather than the number of clients. Figure 2 shows the runtimes to execute $\pi_{\mathsf{ROC}}$ for an increasing number of datapoints, ranging from 10 to 100,000, considered for FL to compute the confusion matrices. We see that for a hundred thousand training samples contributed by clients in the federation, it takes about 28 hours to find the optimal classification thresholds for each sensitive group. We argue that this longer runtime could be acceptable as the thresholds need to be computed only once at the end of the federated model training. The participating clients then split their information (training labels, inferred labels for the training data with the learned model, sensitive attribute values) into secret shares

11. All the runtimes in Figure 1 and 2 are averaged over 5 runs. We simulated the large number of clients and datapoints by making copies of the 109 clients and their training samples that we used for our experiments in the paper.

and send them to the computing parties. The MPC overhead for the clients to contribute to model debiasing is negligible compared to the clients' model training costs. The MPC computational and communication overhead is instead carried by the computing parties who execute the MPC protocol, without requiring continued presence nor active participation of the clients. The computing parties publish DP ROC curves to the FL aggregator who can then compute the thresholds to make the current model available for inference to the clients, at no additional cost to the clients.

For practical deployment, the runtime of our solution can be further improved with optimizations of a cryptographic nature if we leave the MP-SPDZ framework. Secure multiplications are the bottleneck of any MPC-based solution. While we purposefully designed our MPC protocols to reduce the number of multiplications as much as possible, many multiplications remain. These secure multiplications are implemented using Beaver triples. A well known improvement is to pre-compute these triples during an "offline phase" (an option not available in MP-SPDZ) so that they are immediately available for consumption by the MPC protocols for postprocessing during the "online phase". The runtimes reported in our paper include both the pre-computations that can be done during an offline phase (before the computing parties become active) and an online phase (that starts when the clients send encrypted shares of their information to the computing parties). Performing the offline computations prior to model debiasing (e.g. during model training), can make our `PrivFairFL-Post` more efficient for practical use.

**Privacy-Utility-Fairness Trade-Offs of our proposed protocols.** The strength of our proposed techniques is using MPC which provides complete privacy to the *inputs* while maintaining the utility. We note that MPC protocols are supported by mathematical proofs with no privacy loss metrics. We add noise to the aggregated *outputs* using event level DP, which are then post-processed to compute outputs that see almost no variation in the ratio of the weights (PrivFairFL-Pre) and ratio of TPR and FPR (generated for PrivFair-Post). The simplicity of our approach is in secure aggregation of these statistics that aid in bias mitigation while preserving the utility of the model.

We can study the privacy and accuracy/fairness trade-offs only with respect to the privacy that we provide to the *outputs* with respect to the privacy budget $\epsilon$ used in our MPC protocols to provide DP guarantees. With varying values of $\epsilon$ ranging from 0.25 to 50, we see almost no loss in utility or fairness, highlighting the advantages of emulated global DP using secure aggregation. For local DP on the other hand (i.e., MPC-free solution), for high values of $\epsilon$ ranging from 5 to 50, we did observe that the generated weights have large standard deviation, leading to a change of the ratios of the computed statistics that affect the fairness metrics. This is in line with known results that local DP affects utility much more aggressively.

## Appendix F. Limitations and Broader Impact

In this work, we propose a framework to simultaneously achieve fairness and privacy in training a machine learning model in a federated scenario. However, it is important to note that our approach cannot work with all fairness notions. Since fairness notions are context dependent, it is clear that our framework cannot and should not be deployed for every application area.

We acknowledge that our approach and our proposed protocols can be used in ways other than the ways mentioned in this paper. One can imagine that an attribute that should not be used to distinguish individuals could be used as a sensitive attribute, and the work presented in this paper can enable equalizing the performance across such groups in a private manner. Although our approach can produce fair and private predictions, it is still based on a model produced by a machine learning algorithm. Therefore, it becomes essential to understand that the `PrivFairFL` model could also suffer from the same disadvantages as the original model in aspects that we didn't consider in this work, such as explainability, safety, security, and robustness. Hence, the user must be aware of such a system's limitations, especially when using these models to replace people in decision making.