Training Differentially Private Models with Secure Multiparty Computation

Sikha Pentyala[1,2], Davis Railsback[1], Ricardo Maia[3], Rafael Dowsley[4], David Melanson[1], Anderson Nascimento[1], Martine De Cock[1]

***Abstract*—We address the problem of learning a machine learning model from training data that originates at multiple data owners, while providing formal privacy guarantees regarding the protection of each owner's data. Existing solutions based on Differential Privacy (DP) achieve this at the cost of a drop in accuracy. Solutions based on Secure Multiparty Computation (MPC) do not incur such accuracy loss but leak information when the trained model is made publicly available. We propose an MPC solution for training DP models. Our solution relies on an MPC protocol for model training, and an MPC protocol for perturbing the trained model coefficients with Laplace noise in a privacy-preserving manner. The resulting MPC+DP approach achieves higher accuracy than a pure DP approach, while providing the same formal privacy guarantees. Our work obtained first place in the iDASH2021 Track III competition on confidential computing for secure genome analysis.**

***Index Terms*—Secure Multiparty Computation, Differential Privacy, Logistic Regression**

I. Introduction

The ability to induce a machine learning (ML) model from data that originates at multiple data owners while protecting the privacy of each data owner, is of great practical value in a wide range of applications, for a variety of reasons. Most prominently, training on more data typically yields higher quality ML models. For instance, one could train a more accurate model to predict the length of hospital stay of COVID-19 patients when combining data from multiple clinics. This is an application where the data is *horizontally distributed*, meaning that each data owner (clinic) has records/rows of the data. Furthermore, being able to combine different data sets enables new applications that pool together data from multiple data owners, or even from different data owners within the same organization. An example of this is an ML model that relies on lab test results as well as healthcare bill payment information about patients, which are usually managed by different departments within a hospital system. This is an example of an application where the data is *vertically distributed*, i.e. each data owner has their own columns. While there are clear advantages to training ML models over data that is distributed across multiple data owners, in practice often these data owners do not want to disclose their data to each other, because the data in itself constitutes a competitive advantage, or because the data owners need to comply with data privacy regulations.

The importance of enabling privacy-preserving model training has spurred a large research effort in this domain, most notably in the development and use of Privacy-Enhancing Technologies (PETs), prominently including Federated Learning (FL) [1], Differential Privacy (DP) [2], Secure Multiparty Computation (MPC) [3], and Homomorphic Encryption (HE) [4]. Each of these techniques has its own (dis)advantages. Approaches based on (combinations of) FL, MPC, or HE alone do not provide sufficient protection if the trained model is to be made publicly known, or even if it is only made available for black-box query access, because information about the model and its training data is leaked through the ability to query the model [5]–[8]. Formal privacy guarantees in this case can be provided by DP, however at a cost of accuracy loss that is inversely proportional to the *privacy budget* (see Sec. II). To mitigate this accuracy loss, we propose an MPC solution for training DP models.

Our Approach. Rather than having each party training a model in isolation on their own data set, we have the parties running an MPC protocol on the totality of the data sets without requiring each party to disclose their private information to anyone. Since we restrict our analysis to generalized linear models, we then have these parties using MPC to generate the necessary noise and privately adding it to the weights of the trained classifier to satisfy DP requirements. We show that this procedure yield the same accuracy and DP guarantees as in the global DP model however without requiring the parties to reveal their data to a central aggregator, or to anyone else for that matter. Indeed, the MPC protocols effectively play the role of a trusted curator implementing global DP. The resulting classifier can be published in the clear, or used for private inference on top of MPC. Our solution is applicable in scenarios in which the data is horizontally distributed as well as in scenarios where the data is vertically distributed. It obtained the highest accuracy in the iDASH2021 Track III competition for training a model to predict the risk of wild-type transthyretin amyloid cardiomyopathy using medical claims data from two data owners, while providing DP guarantees.

II. Preliminaries

Differential Privacy. DP in general is concerned with providing aggregate information about a data set D without

[1]School of Engineering and Technology, University of Washington, Tacoma, United States; [2]Mila, Quebec AI Institute, Montreal, Canada; [3]Department of Computer Science, University of Brasilia, Brasilia, Brazil; [4]Department of Software Systems and Cybersecurity, Monash University, Melbourne, Australia; Correspondance to Sikha Pentyala (sikha@uw.edu)

disclosing information about specific individuals in D [2]. A data set D' that differs in a single entry from D is called a neighboring database. A randomized algorithm $\mathcal{A}$ is called (ϵ, δ)-DP if for all pairs of neighboring databases D and D', and for all subsets S of $\mathcal{A}$'s range,

$$\mathrm{P}(\mathcal{A}(D) \in S) \leq e^{\epsilon} \cdot \mathrm{P}(\mathcal{A}(D') \in S) + \delta. \quad (1)$$

In other words, $\mathcal{A}$ is DP if $\mathcal{A}$ generates similar probability distributions over outputs on neighboring data sets D and D'. The parameter $\epsilon \geq 0$ denotes the *privacy budget* or privacy loss, while $\delta \geq 0$ denotes the probability of violation of privacy, with smaller values indicating stronger privacy guarantees in both cases. ϵ-DP is a shorthand for $(\epsilon, 0)$-DP. $\mathcal{A}$ can for instance be an algorithm that takes as input a data set D of training examples and outputs an ML model. An (ϵ, δ)-DP randomized algorithm $\mathcal{A}$ is commonly created out of an algorithm $\mathcal{A}^*$ by adding noise that is proportional to the sensitivity of $\mathcal{A}^*$. We describe the Laplace noise technique that we use to this end in detail in Sec. IV.

Secure Multiparty Computation. MPC is an umbrella term for cryptographic approaches that allow two or more parties to jointly compute a specified output from their private information in a distributed fashion, without revealing the private information to each other [3]. MPC is concerned with the protocol execution coming under attack by an adversary which may corrupt one or more of the parties to learn private information or cause the result of the computation to be incorrect. MPC protocols are designed to prevent such attacks being successful, and can be mathematically proven to guarantee privacy and correctness. We follow the standard definition of the Universal Composability (UC) framework [9], in which the security of protocols is analyzed by comparing a real world with an ideal world. For details, see Evans et al. [10].

An adversary can corrupt a certain number of parties. In a *dishonest-majority* setting the adversary is able to corrupt half of the parties or more if he wants, while in an *honest-majority* setting, more than half of the parties are always honest (not corrupted). Furthermore, the adversary can have different levels of adversarial power. In the *semi-honest* model, even corrupted parties follow the instructions of the protocol, but the adversary attempts to learn private information from the internal state of the corrupted parties and the messages that they receive. MPC protocols that are secure against semi-honest or *"passive"* adversaries prevent such leakage of information. In the *malicious* adversarial model, the corrupted parties can arbitrarily deviate from the protocol specification. Providing security in the presence of malicious or *"active"* adversaries, i.e. ensuring that no such adversarial attack can succeed, comes at a higher computational cost than in the passive case. The protocols in Sec. IV 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.

As an illustration, we describe the well-known additive secret-sharing scheme for dishonest-majority 2PC with passive adversaries that we used in the iDASH2021 Track III competition. In Sec.V we additionally present results for honest-majority 3PC and 4PC schemes with passive and active adversaries; for details about those MPC schemes we refer to [11], [12]. In the additive secret-sharing 2PC scheme there are two computing parties, nicknamed *Alice* and *Bob*. All computations are done on integers, modulo an integer q. The modulo q is a hyperparameter that defines the algebraic structure in which the computations are done. A value x in $\mathbb{Z}_q = \{0, 1, \ldots, q-1\}$ is secret shared between Alice and Bob by picking uniformly random values $x_1, x_2 \in \mathbb{Z}_q$ such that $x_1 + x_2 = x \mod q$. x_1 and x_2 are additive shares of x (which are delivered to Alice and Bob, respectively). Note that no information about the secret value x is recovered by any of the individual shares x_1 or x_2, but the secret-shared value x can be trivially revealed by combining both shares x_1 and x_2. The parties Alice and Bob can jointly perform computations on numbers by performing computations on their own shares, without the parties learning the values of the numbers themselves.

For protocols in the passive-security setting, we use $[\![x]\!]$ as a shorthand for a secret sharing of x, i.e. $[\![x]\!] = (x_1, x_2)$. Given secret-shared values $[\![x]\!] = (x_1, x_2)$ and $[\![y]\!] = (y_1, y_2)$, and a constant c, Alice and Bob can jointly perform the following operations, each by doing only local computations on their own shares:[1]

- Addition of a constant ($z = x + c$): Alice and Bob compute $(x_1 + c, x_2)$. Note that Alice adds c to her share x_1, while Bob keeps the same share x_2. This operation is denoted by $[\![z]\!] \leftarrow [\![x]\!] + c$.
- Addition ($z = x + y$): Alice and Bob compute $(x_1 + y_1, x_2 + y_2)$ by adding their local shares of x and y. This operation is denoted by $[\![z]\!] \leftarrow [\![x]\!] + [\![y]\!]$.
- Multiplication by a constant ($z = c \cdot x$): Alice and Bob compute $(c \cdot x_1, c \cdot x_2)$ by multiplying their local shares of x by c. This operation is denoted by $[\![z]\!] \leftarrow c[\![x]\!]$.

Multiplication of secret-shared values $[\![x]\!]$ and $[\![y]\!]$ is done using a so-called *multiplication triple* [13], which is a triple of secret-shared values $[\![u]\!]$, $[\![v]\!]$, $[\![w]\!]$, such that u and v are uniformly random values in $\mathbb{Z}_q$ and $w = u \cdot v$. Given that they have a multiplication triple, Alice and Bob can compute $[\![d]\!] = [\![x]\!] - [\![u]\!]$ and $[\![e]\!] = [\![y]\!] - [\![v]\!]$, and, in a communication step, *open* d and e by disclosing their respective shares of d and e to each other. Next, they can compute $[\![z]\!] = [\![w]\!] + d \cdot [\![v]\!] + e \cdot [\![u]\!] + d \cdot e$, which is equal to $[\![x \cdot y]\!]$. We denote this operation by $[\![z]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x]\!], [\![y]\!])$. Each multiplication requires a fresh multiplication triple. Such triples can be predistributed by a trusted initializer (TI). In case the that a TI is not available or desirable, Alice and Bob can simulate the role of the TI, at the cost of additional pre-processing time and computational assumptions, see [14].

Building on these cryptographic primitives, MPC protocols for other operations can be developed, including for privacy-preserving training of ML models and noise generation to provide DP guarantees (see Sec. IV). Our protocols use the well known subprotocols for division π_{DIV} of secret-shared values, square root π_{SQRT} of secret-shared values, and generation of random values from a uniform distribution $\pi_{\mathsf{GR-RANDOM}}$ from MP-SPDZ [15].

[1] We often omit the modular notation for conciseness.

III. Related Work

Our approach preserves input privacy, i.e., it ensures that the training data sets are not exposed (except under ϵ-DP guarantees) to anyone but their original holders during (1) model training and (2) publication or inference. As we describe below, existing methods either do not fully protect input privacy, or they do so at the cost of higher accuracy loss than our approach.

MPC/HE based Model Training. Many cryptography based methods have been proposed for privacy-preserving learning of ML models with data from multiple data owners, including for linear regression models [16], [17], (ensembles of) decision trees [18]–[21], and neural network architectures [14], [22]–[24]. These techniques protect input privacy during training while still, in principle, producing the same ML models that one would obtain in the clear (i.e. when no encryption is used). The latter is both a blessing, as there is no accuracy loss, and a problem, as upon model publication or during inference, the trained models leak the same kind of information as models trained in-the-clear [5]–[8]. Because these methods do not provide DP guarantees, we do not compare with them in Sec. V.

DP and FL based Model Training. Much of the literature on training DP models [25] is developed for the *global* DP (a.k.a. *central* DP) paradigm, which assumes the existence of a trusted curator (aggregator) who collects all the data and then trains a DP model over it, e.g. by adding noise to the gradients or the model coefficients. These methods do not preserve input privacy, since data owners need to disclose their data sets to the aggregator. A *local* DP approach in which privacy loss is controlled by having the data owners add noise to their input data *before* disclosing it to the aggregator, results in substantial accuracy degradation. In our approach we eliminate the need for a trusted curator by simulating this entity through MPC protocols that are run directly by the parties themselves.

Another related existing approach combines Federated Learning (FL) with DP. In FL, each of the data owners participates in model training on their end and only exchanges trained model parameters or gradients with the central server [1]. To provide DP guarantees, the data owners can add noise to protect the values that they send to the central server. In Sec. V we compare with such an approach in which the data owners perturb their model coefficients before sending them to the central server for aggregation. This approach works only in the horizontally distributed data setting, while our approach (see Sec. IV) works in the vertically distributed setting as well.

Combinations of MPC and DP. The key idea in our proposed approach is to train DP models while performing as much of the computations as possible in MPC protocols in order to preserve accuracy. MPC and DP for ML have been well studied in isolation, but the strong privacy protections that can result from their synergy are still being explored [26]. We combine MPC and DP to protect training data privacy during training *and* during inference. In practice, we simulate the trusted curator present in the centralized DP model by using MPC. While in the past such approach was avoided, due to the high computational cost of training the models on top of MPC, we argue that, with advances in protocols and computing power, the higher utility provided by such approach justifies its adoption in several situations.

Combining MPC with DP has been proposed in the context of FL, where the data is *horizontally* distributed (see e.g. [27]–[29]). No solutions for the *vertically* partitioned scenario exist. Another possible approach is to use cryptographic protocols (not necessarily MPC) and differential privacy, such as in [28]–[32], in order to train individual models on the data sets in possession of the computing parties and aggregate these models by averaging their coefficients. Again, this approach does not work for vertically partitioned data. Moreover, our solution trains the final model on the union of all the individual data sets, thus essentially obtaining the same utility that is achievable in the trusted curator scenario.

IV. Method

A. Overview

We work in the scenario described in Fig. 1 distinguishing between the *data owners* who hold the training data sets on one hand, and the *computing parties* who run the MPC protocols for model training and noise addition. Our solution works in scenarios in which each data owner (e.g. hospital or bank) is also a computing party, as well as in scenarios where the data owners outsource the computations to untrusted servers (computing parties) instead. The data holders secret share their data with a set of computing servers. The servers run an MPC protocol and produce an ML model protected by DP. We implement our solution for 2, 3 and 4 computing servers, but they are general and work with any number of computing servers as well as data holders, by choosing an appropriate underlying MPC scheme for the desired number of computing parties (see Sec. II). The resulting model can be used for private inference (on top of the underlying MPC protocol) or made open to the public.

The core of our solution is an MPC protocol (Sec. IV-C) implementing a mechanism for providing ϵ-DP by perturbing the coefficients of a trained logistic regression (LR) model with the addition of a noise vector η that is sampled according to the density function

$$h(\eta) \propto e^{-\frac{n\epsilon\Lambda}{2}\|\eta\|} \tag{2}$$

[33], [34]. In the above expression for the density function, n is the number of instances that were used to train the LR model, and Λ is the regularization parameter (regularization strength) used during training. This technique provides ϵ-DP provided that:

(C1) each input feature vector has an L2 norm of at most 1;
(C2) the LR model is trained using L2 regularization.

If the preceding conditions are fulfilled, then the sensitivity of LR with regularization parameter Λ is at most $\frac{2}{n\Lambda}$ [33], [34].

In all MPC protocols used in this paper, secret sharings are in $\mathbb{Z}_q$ with $q = 2^\lambda$, i.e. a power 2 ring. In Sec. V we present results with $\lambda = 64$ for a varying number of data owners, and for 2, 3, and 4 computing parties. Since all computations in MPC are done over integers in $\mathbb{Z}_q$ (see Sec. II), the data owners first convert the real numbers in their data to integers using a fixed-point representation [35] and subsequently split the integer

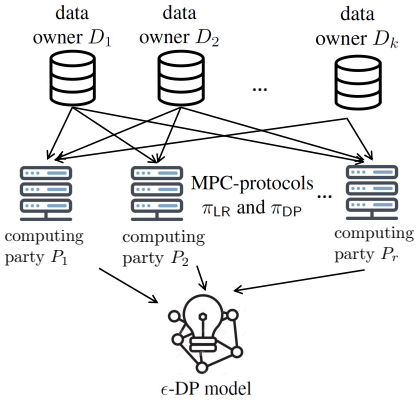

Fig. 1. Privacy-preserving training of an ϵ-DP model with MPC

values into secret shares which are sent to the computing parties (see Fig. 1). While the original value of a secret-shared number can be trivially revealed by combining the shares, the secret-sharing based MPC schemes ensure that nothing about the inputs is revealed to any subset of the computing parties that can be corrupted by an adversary. This means, in particular, that no computing party by itself learns anything about the actual values of the inputs. Next, the computing parties proceed by performing computations on the shares. In particular, the computing parties:

1) Jointly run MPC protocol π_{LR} (Sec. IV-B) to L2 normalize the training data, and to subsequently induce a LR model using L2 regularization from the normalized data. At the end of this protocol, the coefficients of the model are secret-shared between the parties.
2) Jointly run MPC protocol π_{DP} (Sec. IV-C) to add a noise vector to the secret-shared model coefficients. At the end of this protocol, the noisy coefficients of the model are secret shared between the parties.
3) Disclose their shares of the LR coefficients so that they can be combined in a final ϵ-DP LR model.

As the noise in step 2 is generated and added to the weights using MPC, the computing parties will not learn it, hence they will not be able to retrieve the actual model coefficients from the noisy coefficients that are disclosed in step 3.

B. Protocol π_{LR} for Model Training

Protocol π_{LR} is based on an existing implementation of an MPC protocol for training a LR with SGD optimizer that is available in MP-SPDZ [15]. We have extended this protocol in two ways to satisfy the conditions **(C1)** and **(C2)** for the mechanism from [33] to provide ϵ-DP. Pseudocode for π_{LR} is presented in Prot. 1. At the beginning of the protocol, the computing parties have secret shares of a set of labeled training examples. To satisfy condition **(C1)**, on Line 1–3 the computing parties first apply L2 normalization to the secret shares of each training example by running protocol π_{NORM}; pseudocode for π_{NORM} is provided separately in Prot. 2.

The computing parties then begin secure training on the privately L2 normalized data from all the data owners. Lines 4–9 present an overview of an existing MPC protocol for training a LR model with SGD optimizer in MP-SPDZ. The

Algorithm 1: Protocol π_{LR} for secure logistic regression training

1 **Input:** A set $S = \{([\![\mathbf{x}]\!], [\![t]\!])\}$ of secret-shared training examples, each consisting of a secret-shared input feature vector $\mathbf{x}$ of length m and a secret shared label t; learning rate α; regularization penalty Λ; momentum C; number of iterations n_{iter}; batch size B.
2 **Output:** A secret-shared vector $[\![\mathbf{w}]\!]$ of weights w_i that minimize the sum of squared errors over the training data
1: **for all** training examples $([\![\mathbf{x}]\!], [\![t]\!])$ in S **do**
2: $\quad [\![\mathbf{x}]\!] \leftarrow \pi_{\mathsf{NORM}}([\![\mathbf{x}]\!], m)$
3: **end for**
4: $[\![\mathbf{w}]\!] \leftarrow \pi_{\mathsf{INIT}}$ {MP-SPDZ module for Glorot uniform initializer}
5: **for** $i = 1$ **to** n_{iter} **do**
6: $\quad$ **for** $j = 1$ **to** B **do**
7: $\quad\quad$ Run π_{FWD} {MP-SPDZ module for forward pass}
8: $\quad\quad$ Run π_{BKWD} {MP-SPDZ module for backward pass}
9: $\quad$ **end for**
10: $\quad$ Run π_{UPDATE} {Modified MP-SPDZ module for weight updates} with the modified update rule for computing $\Delta\mathbf{w}$: $[\![\Delta\mathbf{w}]\!] \leftarrow C[\![\Delta\mathbf{w}]\!] - \alpha[\![\Delta\mathbf{w}]\!] - \Lambda\alpha[\![\mathbf{w}]\!]$
11: **end for**
12: **return** $[\![\mathbf{w}]\!]$

Algorithm 2: Protocol π_{NORM} for secure $L2$ normalization

1 **Input:** A secret-shared vector $[\![\mathbf{x}]\!]$ of length d
2 **Output:** Secret-shared L2 normalized vector $[\![\mathbf{x}^{\mathbf{norm}}]\!]$
1: Declare vector $[\![\mathbf{x}^{\mathbf{norm}}]\!]$ of length d
2: $[\![S]\!] \leftarrow 0$
3: **for** $i = 1$ **to** d **do**
4: $\quad [\![S]\!] \leftarrow [\![S]\!] + \pi_{\mathsf{MUL}}([\![x_i]\!], [\![x_i]\!])$
5: **end for**
6: $[\![v]\!] = \pi_{\mathsf{DIV}}(1, \pi_{\mathsf{SQRT}}([\![S]\!]))$
7: **for** $i = 1$ **to** d **do**
8: $\quad [\![x_i^{norm}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![x_i]\!], [\![v]\!])$
9: **end for**
10: **return** $[\![\mathbf{x}^{\mathbf{norm}}]\!]$

training begins with initializing the secret shares of the weights (coefficients) of the LR model using Glorot uniform initializer [36]. To this end, the computing parties execute protocol π_{INIT} on Line 4. The training is carried out for n_{iter} number of iterations, which is a public constant agreed upon by all computing parties along with the learning rate α, the regularization penalty Λ, the momentum C, and the batch size B. The training is done in batches (Lines 6–9). For each batch, the MP-SPDZ module π_{FWD} for a secure forward pass is called on Line 7, followed by the MP-SPDZ module π_{BKWD} for a backward pass on Line 8. The secret shares of the weights are then updated for each batch using the MP-SPDZ module for updating the weights. We modified this module to satisfy **(C2)** with L2 regularization as per Line 10 in Prot. 1.

If the data is horizontally distributed across the data owners, then each data owner can apply sample-wise L2 normalization to their own instances before secret sharing the training instances with the computing parties. The computing parties in this case can skip Line 1–3 of Prot. 1, which will reduce the training runtime.

C. Protocol π_{DP} for Noise Generation

At the end of MPC protocol π_{LR}, the coefficients $\mathbf{w}$ of the trained LR model are secret-shared between the parties. Next, the parties run the MPC protocol π_{DP}, described in this section, to generate noise and add it to the model coefficients

Algorithm 3: Protocol π_{DP} for secure output perturbation

Input: A secret-shared vector $[\![\mathbf{w}]\!]$ with d model coefficients w_i; regularization penalty Λ; total number n of training examples; privacy budget ϵ.
Output: secret-shared vector $[\![\widetilde{\mathbf{w}}]\!]$ with perturbed model coefficients

1: $[\![\mathbf{s}]\!] \leftarrow \pi_{\mathsf{GSS}}(d)$
2: $[\![\mathbf{s}]\!] \leftarrow \pi_{\mathsf{NORM}}([\![\mathbf{s}]\!], d)$
3: $[\![\gamma]\!] \leftarrow \pi_{\mathsf{GMS}}(n, d, \epsilon, \Lambda)$
4: Initialize vector $[\![\widetilde{\mathbf{w}}]\!]$ of length d to $[\![\mathbf{0}]\!]$
5: **for** $i = 1$ **to** d **do**
6: $\quad [\![s_i]\!] \leftarrow \pi_{\mathsf{MUL}}([\![s_i]\!], [\![\gamma]\!])$
7: $\quad [\![\widetilde{w}_i]\!] \leftarrow [\![w_i]\!] + [\![s_i]\!]$
8: **end for**
9: **return**$[\![\widetilde{\mathbf{w}}]\!]$

to provide DP guarantees. Protocol π_{DP} implements the output perturbation method (or sensitivity method) [33], [34] in a privacy-preserving way. While the original output perturbation method relies on the fact that the model coefficients are known or disclosed to a single entity, such as a trusted curator, we do not make such an assumption. Instead, the model coefficients remain secret-shared among the computing parties, neither of which knows the true values. The challenge is for the parties to jointly generate noise that is appropriate for the true model coefficients that they cannot see, without learning the true value of the noise. Indeed, no entity should learn the true value of the noise, so that the noisy model coefficients can safely be disclosed at the end of the process (see step 3 in Sec. IV-A), without leaking information that would violate the DP guarantees.

We proceed by describing the output perturbation method in the clear [33], [34] followed by a more detailed description of our MPC protocol π_{DP}. In the output perturbation method, sensitivity is defined using the L2 norm, and the noise vector is sampled from a particular instance of a multidimensional power exponential distribution $h(\eta) \propto e^{-\frac{n\epsilon\Lambda}{2}\|\eta\|}$. Following [33], [37], we can sample a vector $\mathbf{s}$ according to the distribution $h(\eta)$, by following these steps in which d is the length of the vector (i.e. the number of model coefficients):

(O1) Generate a d-dimensional Gaussian vector $\mathbf{s}$. That is, each coordinate of the vector is sampled from a Gaussian distribution with mean zero and variance one.
(O2) Normalize $\mathbf{s}$, that is divide each coordinate of $\mathbf{s}$ by its L2 norm.
(O3) Sample the gamma distribution $\Gamma(d, \frac{2}{n\epsilon\Lambda})$ to obtain a value γ, and multiply each coordinate of the normalized vector produced in step 2 with γ.

Steps (O1)-(O2) generate a random d-dimensional vector on the unit sphere (this follows from the spherical symmetry of the multivariate Gaussian distribution). Step (O3) changes the magnitude of this vector to an appropriate value. The obtained vector is then added to the vector of model coefficients.

Prot. 3 presents pseudocode for MPC protocol π_{DP} that the computing parties should execute jointly to perturb the secret-shared model coefficients $[\![\mathbf{w}]\!]$ according to the output perturbation method, in a fully privacy-preserving way. At the beginning of π_{DP}, the parties have a secret-shared vector $[\![\mathbf{w}]\!]$ of the model coefficients. On Line 1, the parties run protocol π_{GSS} to generate a secret-shared vector $[\![\mathbf{s}]\!]$ of length d sampled from

Algorithm 4: Protocol π_{GSS} for secure sampling of a vector from a Gaussian distribution

Input: Vector length d.
Output: A secret-shared vector $[\![\mathbf{s}]\!]$ of length d sampled from Gaussian distribution with mean 0 and variance 1

1: Declare vector $[\![\mathbf{s}]\!]$ of length d
2: **for** $i = 0$ **to** $d/2$ **do**
3: $\quad [\![u]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$
4: $\quad [\![v]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$
5: $\quad [\![r]\!] \leftarrow \pi_{\mathsf{SQRT}}(-2\pi_{\mathsf{LN}}([\![u]\!]))$
6: $\quad [\![\theta]\!] \leftarrow 2\pi[\![v]\!]$
7: $\quad [\![s_{2i}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![r]\!], \pi_{\mathsf{COS}}([\![\theta]\!]))$
8: $\quad [\![x_{2i+1}]\!] \leftarrow \pi_{\mathsf{MUL}}([\![r]\!], \pi_{\mathsf{SIN}}([\![\theta]\!]))$
9: **end for**
10: **if** d is odd **then**
11: $\quad [\![p]\!] \leftarrow \pi_{\mathsf{GSS}}(2)$
12: $\quad [\![s_{d-1}]\!] \leftarrow [\![p_0]\!]$
13: **end if**
14: **return**$[\![\mathbf{s}]\!]$

a Gaussian distribution, which they subsequently normalize on Line 2 with the protocol π_{NORM} (see Sec. IV-B). Next, on Line 3, the parties run protocol π_{GMS} to sample a secret-shared value $[\![\gamma]\!]$ from the gamma distribution. On Line 5–8 the parties multiply each coordinate of $\mathbf{x}$ with γ to obtain the appropriate magnitude, and add the obtained noise to the corresponding model coefficient.

Below we describe the protocols π_{GSS} and π_{GMS} for sampling of the noise on Line 1 and 3 in Prot. 3 respectively. The importance of these protocols stems from the fact that they enable the parties to generate secret shares of noise, without each party learning the true value of the noise that they add to the model coefficients in Line 7 of Prot. 3.

Description of protocol π_{GSS}. Prot. 4 relies on the Box-Muller transform [38] to generate samples of the Gaussian unitary distribution, namely $\lceil d/2 \rceil$ pairs of Gaussian samples. For each pair, on Line 3–4 the computing parties securely generate secret shares of two random numbers u and v uniformly distributed in [0,1] by executing $\pi_{\mathsf{GR-RANDOM}}$. In $\pi_{\mathsf{GR-RANDOM}}$, each party generates l random bits, where l is the fractional precision of the power 2 ring representation of real numbers, and then the parties define the bitwise XOR of these l bits as the binary representation of the random number jointly generated. On Line 5–8, the parties then jointly compute a secret sharing of $\sqrt{-2\ln(u)} \cdot \cos(2\pi v)$ and of $\sqrt{-2\ln(u)} \cdot \sin(2\pi v)$ using existing MPC protocols π_{SQRT}, π_{SIN}, π_{COS}, and π_{LN} [15]. In case d is odd, one more sample needs to be generated. The parties do so on Line 11–12 by executing π_{GSS} to sample a vector of length 2 and only retain the first coordinate.

Description of protocol π_{GMS}. In π_{GMS}, pseudocode for which is provided in Prot. 5, to generate a secret-shared sample $[\![\gamma]\!]$ from the $\Gamma(d, \frac{2}{n\epsilon\Lambda})$ distribution, on Line 1–6, the computing parties generate secret shares of d independent samples from the exponential distribution with rate parameter one (here denoted by Exp(1)) and add them. To generate secret shares of one such sample we use the inverse transform sampling over MPC, which consists of computing $-\ln u$, where u is a random number with precision equal to l bits generated by the computing parties within the interval $[0, 1]$:

Algorithm 5: π_{GMS} for secure sampling from Gamma distribution

Input: Vector length d; total number n of training examples; regularization penalty Λ; privacy budget ϵ
Output: Secret-shared value $[\![\gamma]\!]$ sampled from $\Gamma(d, \frac{2}{n\epsilon\Lambda})$
1: $[\![\gamma]\!] \leftarrow [\![0]\!]$
2: **for** $i = 1$ **to** d **do**
3: $\quad [\![u]\!] \leftarrow \pi_{\mathsf{GR-RANDOM}}(0, 1)$
4: $\quad [\![u]\!] \leftarrow -1\pi_{\mathsf{LN}}([\![u]\!])$
5: $\quad [\![\gamma]\!] \leftarrow [\![\gamma]\!] + [\![u]\!]$
6: **end for**
7: $c \leftarrow 2/(n \cdot \epsilon \cdot \Lambda)$
8: $[\![\gamma]\!] \leftarrow c[\![\gamma]\!]$
9: **return** $[\![\gamma]\!]$

1) On Line 3 the parties execute $\pi_{\mathsf{GR-RANDOM}}$ as in Prot. 4 to generate a random number with precision l in $[0, 1]$. Denote this number by u.
2) On Line 4 the parties compute secret shares of $-\ln(u)$.

Finally, on Line 7–8 the parties scale the sum by multiplying the secret shares with the factor $\frac{2}{n\epsilon\Lambda}$.

The correctness of the protocol follows from the correctness of the inverse transform sampling algorithm, and the fact that $\text{Exp}(1) = \Gamma(1, 1)$ and that $\sum_{i=1}^{d} \Gamma(1, 1) = \Gamma(d, 1)$. Moreover, it follows from the definition of the Gamma distribution that $c\Gamma(d, 1) = \Gamma(d, c)$.

The security of the whole protocol follows from the security guarantees provided by MP-SPDZ [15].

V. Results

A. iDASH 2021 Competition Results

We submitted our approach to a competition hosted in 2021 by iDASH, a National Center for Biomedical Computing funded by the NIH. In Track III of the iDASH 2021 competition, participants were invited to submit solutions for learning a ML model from training data hosted by two virtual centers, while providing DP guarantees. The centers represent data owners who have medical records of respectively 831 patients and 882 patients. Both data sets have the same schema, consisting of 1,874 boolean input attributes and a boolean target variable. The goal is to train a classifier for diagnosis of transthyretin amyloid cardiomyopathy using medical claims data [39]. Solutions submitted to the competition were required to run on two machines. They were evaluated in terms of (1) training runtime on two nodes with Intel Xeon E3-1280 v5 processors (4 physical cores, hyper-threading enabled) and 64 GiB memory; (2) accuracy on a held-out test of 429 patients.

Table I contains the results for the best performing teams satisfying the ϵ-DP requirement (with ϵ set as 3 by the organizers). The first row corresponds to the approach presented in Sec. IV. We implemented the π_{LR} and π_{DP} protocols in MP-SPDZ, an open source framework for MPC [15]. As the underlying MPC scheme for the iDASH2021 competition, we used semi2k (a semi-honest adaptation of [40]) with mixed circuits that employ techniques using secret random bits (extended doubly-authenticated bits; edaBits) [41]. The implementation of this MPC scheme in MP-SPDZ enables secure 2PC against semi-honest adversaries and complied with the requirements of the competition. As the regularizer for LR training, we used $N(\mathbf{w}) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w}$, in which $\mathbf{w}$ denotes the vector of weights (coefficients) of the LR model, i.e. we used $\Lambda = 1$.

All methods in Table I provide ϵ-DP guarantees, with $\epsilon = 3$. The differences among the methods are in the utility (accuracy) and in the time taken to train a DP model. Our π_{LR} +π_{DP} approach achieved the highest accuracy of all methods, while taking the longest time to complete. Indeed, the runtime for the π_{LR} +π_{DP} approach is orders of magnitude larger than the runtimes for the other methods. This is because the π_{LR} +π_{DP} approach is the only method in Table I that uses MPC, while the other methods do not rely on cryptographic techniques. Approach 2 was based on feature selection and training an ensemble of LR models on selected feature subsets, while approach 4 was based on training a decision tree in a DP manner; these approaches were not created by us, and, to the best of our knowledge, their description has not been published in the open literature. In addition to the method from Sec. IV we submitted an MPC-free baseline method to iDASH2021. We describe this method, which corresponds to approach 3 in Table I, below as we also use it for further analysis and comparison in Sec. V-B.

Baseline Method. The baseline technique follows a FL setup with horizontally distributed data in which each data owner locally trains a model on their own data and adds noise to the model parameters at their end. Each data owner then shares its noisy parameters with a central server who performs averaging of the noisy model parameters and sends the result to the data owners. At the end of this process, each data owner holds the aggregated trained model. In more detail, in the baseline technique, each data owner:

1) Applies L2 normalization to its own instances;
2) Trains a LR model on its normalized instances;[2]
3) Adds noise to the trained LR coefficients as per steps (O1)-(O3) in Sec. IV-C.

After going through steps 1-3, the data owners can each publish their perturbed LR coefficients, which we subsequently average to create a final model. Because steps 1–3 provide ϵ-DP [33], [34], and since the data sets do not have common entries (a case of parallel composition), the overall solution provides ϵ-DP due to the post-processing property of differential privacy.

B. Further Analysis

Utility. The baseline technique is only applicable when the data is horizontally distributed among the data owners, while our π_{LR} +π_{DP} approach works in the vertically distributed scenario as well. We argue that even in the horizontally distributed scenario, the π_{LR} +π_{DP} approach is preferable because it yields a higher accuracy, which becomes even more evident when the data is distributed among multiple data owners. To demonstrate this, we perform repeated stratified splits on the iDASH data to distribute it across 4 and 8 data owners. To measure accuracy, we use 5-fold CV where in each fold we hold out 20% of the data of each data owner for testing, and used ϵ-DP with $\epsilon = 1$.

As the last column in Table II shows, the accuracy of our π_{LR} +π_{DP} approach is independent of the number of data owners,

[2]We used the LR implementation from sklearn for this with penalty='l2' (L2 regularization) and $C = 1$ (the inverse of Λ).

TABLE I
IDASH2021 COMPETITION RESULTS FOR ϵ-DP WITH $\epsilon = 3$ AND DATA ORIGINATING FROM TWO DATA OWNERS.

	APPROACH	PETS	ACCURACY[3]	RUNTIME[3]
1.	π_{LR}+π_{DP} (SEC. IV)	MPC & DP	86.25%	$\sim$ 15,000 SEC
2.	FEAT. SEL. AND LR ENSEMBLE	DP	85.31%	31.942 SEC
3.	BASELINE (SEC. V-A)	DP	84.85%	0.27 SEC
4.	DECISION TREE BASED	DP	84.38%	0.09 SEC

TABLE II
ACCURACY RESULTS FOR VARYING NUMBER OF DATA OWNERS FOR ϵ-DP WITH $\epsilon = 1$.

# DATA OWNERS	BASELINE	π_{LR} +π_{DP}
2	85.79%	87.45%
4	83.36%	87.45%
8	76.92%	87.45%

TABLE III
RUNTIMES WITH DIFFERENT NUMBER r OF COMPUTING PARTIES, USING DIFFERENT UNDERLYING MPC SCHEMES.

r	SECURITY	RUNTIME	MPC SCHEME
2	PASSIVE	12951.40 SEC	[40]
3	PASSIVE	75.30 SEC	[11]
3	ACTIVE	847.10 SEC	[12]
4	ACTIVE	128.30 SEC	[12]

as regardless of the number of data owners, the computing parties still train a model over all the training data with π_{LR} and subsequently add noise once to the globally trained model coefficients with π_{DP}, effectively simulating the global DP paradigm but without the involvement of a trusted curator. The baseline technique on the other hand adheres to the local DP paradigm in which each data owner adds noise to its local model. This requires a split of the overall privacy budget among all data owners and results in more noise in the final aggregated model. Furthermore, the utility of our π_{LR} +π_{DP} approach is independent of the number of instances and/or features owned by each individual data owner, while the accuracy of the baseline technique degrades when individual data owners do not have sufficient instances to train local models that generalize well. This is especially relevant in biomedical applications that are characterized by high-dimensional data sets with relatively few instances.

The π_{LR} +π_{DP} approach yields the same accuracy irrespective of the number of computing parties and data owners.

Runtime. As Table III shows, the number of computing parties, the corruption threshold, and respective MPC schemes do have a substantial effect on the training time. The experiments for Table III were run with the same data as in Sec. V-A on co-located F32s V2 Azure virtual machines each of which contains 32 cores, 64 GiB of memory, and network bandwidth of upto 14 Gb/s. Every computing party ran on separate VM instance (connected with a Gigabit Ethernet network). The times reported include computing as well as communication times. The training was run for 300 epochs with batch size of 128, with $\epsilon = 1$, $\Lambda = 1$ and with edaBits for mixed circuit computations.

As expected, we observe that the corruption threshold has the most effect on the run time. Protocols that are secure for an honest majority of players (the protocols presented in [11], and [12]) are much faster than protocols secure against a dishonest majority [40]. We can also observe that, for the same corruption threshold, protocols secure against passive adversaries are faster than protocols secure against active adversaries. The four party protocol proposed in [12] manages to obtain good run times for the case of active adversaries by further reducing the corruption threshold to 25%, i.e. one player out of four can be corrupted by an adversary and the protocol is still secure.

Our results show that MPC implementations for honest majority in the case of realistic sized data sets for genetic studies (a few hundred patients, and a few thousand features) are practical. We can train such models and add DP guarantees on top of MPC in less than 1.5 min for the case of honest majority protocols with passive security. Even in the case of stronger adversarial models, the training can be finished in a few hours – which is still practical for many applications where the increased accuracy payoff is valuable.

VI. CONCLUSION

In this paper, we described a practical and efficient adaptation of distributed logistic regression learning — adding tractable privacy guarantees against model inversion in the absence of a trusted curator which, in real-world scenarios, is often impractical, undesirable, or forbidden. This work led to a 1st place in Track III of the iDASH 2021 Genome Privacy competition. Despite having been formulated for a competition task, the design is intentionally general — it makes no assumption about the data partitioning scenario, number of computing parties / data owners, or the security setting in which it is applied. On the basis of linearity, π_{LR} is interchangeable with all linear learners with no need to reevaluate noise variance. Moreover, the exponential noise mechanism is straightforward to replace by the Gaussian mechanism for scenarios where (ϵ, δ)-DP guarantees are acceptable. As such, the MPC+DP method can be harnessed for model training across a broad range of use cases without requiring extensive tuning by privacy experts.

The proposed approach effectively offers the advantages of global DP but without the involvement of a trusted curator,

[3]As provided by the iDASH2021 competition organizers.

because this curator is simulated by an MPC protocol instead. The trade-off between this MPC for global DP approach and the baseline federated method with local DP can be summarized as operating cost (or running time) versus model accuracy. We empirically showed the added utility of collaborative learning with MPC over the standard federated approach. The effect is particularly apparent as the number of disjoint collaborators grows. We also remark that the baseline method, and existing methods that combine MPC with DP in FL, cannot be applied in cases where data is vertically partitioned which is a commonly-found scenario in medicine and advertising. As such, our MPC+DP method allows collaboration in a strictly larger space of applications. Based on performance results, our protocol is extensible to larger data sets while remaining in a realistic time span for model learning, but could be improved further by custom protocol implementations or given the existence of a *correlated randomness dealer* in suitable scenarios. To further improve upon accuracy, a probable research direction is to introduce MPC protocols for feature selection [42] in both horizontal and vertical partitioning schemes.

Acknowledgements. The authors would like to thank Marcel Keller for making the MP-SPDZ framework available. The authors would like to thank Microsoft for the generous donation of cloud computing credits through the UW Azure Cloud Computing Credits for Research program. Funding support for project activities has been provided by Facebook Research Award for Privacy Enhancing Technologies and partially by The University of Washington Tacoma's Founders Endowment fund.

REFERENCES

[1] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, M. Raykova, H. Qi, D. Ramage, R. Raskar, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," https://arxiv.org/abs/1912.04977, 2021.

[2] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2014.

[3] R. Cramer, I. Damgard, and J. Nielsen, *Secure Multiparty Computation and Secret Sharing*. New York: Cambridge University Press Print, 2015.

[4] K. E. Lauter, "Private ai: Machine learning on encrypted data." *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 324, 2021.

[5] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015, pp. 1322–1333.

[6] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX Security Symposium*, 2016, pp. 601–618.

[7] C. Song, T. Ristenpart, and V. Shmatikov, "Machine learning models that remember too much," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 587–601.

[8] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *28th USENIX Security Symposium*, 2019, pp. 267–284.

[9] R. Canetti, "Security and composition of multiparty cryptographic protocols," *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, 2000.

[10] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Foundations and Trends in Privacy and Security*, vol. 2, no. 2-3, pp. 70–246, 2018.

[11] T. Araki, J. Furukawa, Y. Lindell, A. Nof, and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 805–817.

[12] A. Dalskov, D. Escudero, and M. Keller, "Fantastic four: Honest-majority four-party secure computation with malicious security," in *30th USENIX Security Symposium*, 2021.

[13] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology — CRYPTO '91*, J. Feigenbaum, Ed. Springer Berlin Heidelberg, 1992, pp. 420–432.

[14] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 19–38.

[15] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, p. 1575–1590.

[16] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 345 – 364, 2017.

[17] A. Agarwal, R. Dowsley, N. D. McKinney, D. Wu, C.-T. Lin, M. De Cock, and A. C. A. Nascimento, "Protecting privacy of users in brain-computer interface applications," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 8, pp. 1546–1555, 2019.

[18] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, 2000, pp. 36–54.

[19] S. de Hoogh, B. Schoenmakers, P. Chen, and H. op den Akker, "Practical secure decision tree learning in a teletreatment application," in *Intern. Conf. on Financial Cryptography and Data Security*. Springer, 2014, pp. 179–194.

[20] M. Abspoel, D. Escudero, and N. Volgushev, "Secure training of decision trees with continuous attributes," *Proceedings on Privacy Enhancing Technologies*, no. 1, pp. 167–187, 2021.

[21] S. Adams, C. Choudhary, M. De Cock, R. Dowsley, D. Melanson, A. C. Nascimento, D. Railsback, and J. Shen, "Privacy-preserving training of tree ensembles over continuous data," *Proceedings on Privacy Enhancing Technologies*, no. 2, 2022.

[22] S. Wagh, D. Gupta, and N. Chandran, "SecureNN: 3-party secure computation for neural network training," *Proceedings on Privacy Enhancing Technologies*, no. 3, pp. 26–49, 2019.

[23] C. Guo, A. Hannun, B. Knott, L. van der Maaten, M. Tygert, and R. Zhu, "Secure multiparty computations in floating-point arithmetic," *arXiv:2001.03192*, 2020.

[24] M. De Cock, R. Dowsley, A. C. A. Nascimento, D. Railsback, J. Shen, and A. Todoki, "High performance logistic regression for privacy-preserving genome analysis," *BMC Medical Genomics*, vol. 14(23), 2021.

[25] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Procedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 308–318.

[26] S. Wagh, X. He, A. Machanavajjhala, and P. Mittal, "DP-cryptography: marrying differential privacy and cryptography in emerging applications," *Communications of the ACM*, vol. 64, no. 2, pp. 84–93, 2021.

[27] A. Acar, Z. B. Celik, H. Aksu, A. S. Uluagac, and P. McDaniel, "Achieving secure and differentially private computations in multiparty settings," in *IEEE Symposium on Privacy-Aware Computing*, 2017, pp. 49–59.

[28] B. Jayaraman, L. Wang, D. Evans, and Q. Gu, "Distributed learning without distress: privacy-preserving empirical risk minimization," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 6346–6357.

[29] M. A. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *Advances in Neural Information Processing Systems 23*, 2010, pp. 1876–1884.

[30] M. Chase, R. Gilad-Bachrach, K. Laine, K. Lauter, and P. Rindal, "Private collaborative neural network learning," *Cryptology ePrint Archive*, 2017.

[31] D. Byrd and A. Polychroniadou, "Differentially private secure multi-party computation for federated learning in financial applications," *arXiv preprint arXiv:2010.05867*, 2020.

[32] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou, "A hybrid approach to privacy-preserving federated learning," in *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, 2019, pp. 1–11.

[33] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression." in *Advances in Neural Information Processing Systems*, 2008, pp. 289–296.
[34] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization." *Journal of Machine Learning Research*, vol. 12, no. 3, 2011.
[35] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *14th International Conference on Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 6052. Springer, 2010, pp. 35–50.
[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.
[37] E. G. Sánchez-Manzano, M. A. Gomez-Villegas, and J.-M. Marín-Diazaraque, "A matrix variate generalization of the power exponential family of distributions," *Communications in Statistics-Theory and Methods*, vol. 31, no. 12, pp. 2167–2182, 2002.
[38] G. Box and M. E. Muller, "A note on the generation of random normal deviates," *Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611, 1958.
[39] A. Huda, A. Castaño, A. Niyogi, J. Schumacher, M. Stewart, M. Bruno, M. Hu, F. S. Ahmad, R. C. Deo, and S. J. Shah, "A machine learning model for identifying patients at risk for wild-type transthyretin amyloid cardiomyopathy," *Nature communications*, vol. 12, no. 1, pp. 1–12, 2021.
[40] R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing, "SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for dishonest majority," in *Annual International Cryptology Conference*. Springer, 2018, pp. 769–798.
[41] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl, "Improved primitives for mpc over mixed arithmetic-binary circuits," Cryptology ePrint Archive, Report 2020/338, 2020.
[42] X. Li, R. Dowsley, and M. De Cock, "Privacy-preserving feature selection with secure multiparty computation," *arXiv preprint arXiv:2102.03517*, 2021.
[43] H. Xie, J. Li, Q. Zhang, and Y. Wang, "Comparison among dimensionality reduction techniques based on random projection for cancer classification," *Computational Biology and Chemistry*, vol. 65, pp. 165–172, 2016.

VII. APPENDIX

A. Effect of Privacy Budget ϵ on Accuracy of Models Trained with π_{LR} +π_{DP}

Table IV shows the effect of the privacy budget ϵ on the accuracy of models trained with the π_{LR} +π_{DP} approach. The accuracy is measured over one of the folds of the train and test data from Sec. V-B. The training is done for 300 epochs with batch size of 128 and $\Lambda = 1$. The results are as expected, with a larger privacy budget – i.e. less stringent privacy requirements – yielding more accurate models. The observation that the accuracy for $\epsilon = 1$ is at par with the accuracy for ϵ =INF (i.e. when no noise is added) is explained by the fact that adding some noise can positively impact the generalization capability of the model.

TABLE IV
ACCURACY OF MODELS TRAINED WITH π_{LR} +π_{DP} FOR DIFFERENT VALUES OF ϵ

ϵ	ACCURACY
0.001	52.90%
0.01	70.06%
0.1	83.72%
0.5	84.60%
1	85.67%
INF	85.46%

B. Comparison with Objective Function Perturbation

For the π_{LR} +π_{DP} approach (Sec.IV) and the baseline technique (Sec. V-A), we adopted the sensitivity method that perturbs the model coefficients, i.e. the output perturbation method that was proposed as Algorithm 1 in [34]. We ran experiments with Algorithm 2 from [34] that adds noise to the objective function itself.[3] In the BASELINE-OP method in Table V, each data owner trains a differentially private LR model locally by perturbing the objective function, i.e. using the objective function perturbation method [34]. The resultant coefficients of the local models are then averaged resulting in a final DP model.

As can be seen in Table V, contrary to what one would expect based on the analysis in [34], the accuracy results with this objective function perturbation method were not good on the iDASH2021 data, and far worse than those with the output perturbation method. We attribute this to the high-dimensional nature of the iDASH2021 data (many features and relatively few instances) which is very different from the data sets used for evaluation in [34].

TABLE V
ACCURACY RESULTS OBTAINED WITH 5-FOLD CV FOR ϵ-DP WITH $\epsilon = 1$ AND 2 DATA OWNERS

	APPROACH	ACCURACY
OUTPUT PERTURBATION	π_{LR} +π_{DP} (SEC. IV)	87.45%
	BASELINE (SEC. V-A)	85.79%
OBJECTIVE PERTURBATION	BASELINE-OP	49.40%

C. Experiments on Other Data Sets

We further evaluate our approach on the BC-TCGA and GSE2034 data sets of the iDASH 2019 competition.[4] Both data sets contain gene expression data from breast cancer patients which are normal tissue/non-recurrence samples (negative) or breast cancer tissue/recurrence tumor samples (positive) [43]. We perform experiments with 5-fold CV, where in each fold the training data is distributed between 2 data owners.

a) GSE2034: Each instance in this data set is characterized by 12,634 continuous input attributes and a boolean target variable. There are 895 instances in total. In each iteration of the 5-fold CV, each data owner owns 447-448 instances, 20% of which are held out for testing.

b) BC-TCGA: Each instance in this data set is characterized by 17,814 continuous input attributes and a boolean target variable. There are 1,875 instances in total. In each iteration of the 5-fold CV, each data owner owns 937-938 instances, 20% of which are held out for testing.

For both data sets, the secure training is run for 150 epochs with a batch size of 32, and $\Lambda = 1$. Table VI shows accuracy results obtained with 5-fold CV. To appreciate the inherent difference in difficulty between the GSE2034 and the BC-TCGA classification tasks, as the first results column in Table VI we include the accuracies obtained with a model trained in the

[3]We implemented this approach using IBM's Diffprivlib library https://github.com/IBM/differential-privacy-library.

[4]http://www.humangenomeprivacy.org/2019/competition-tasks.html

central learning paradigm, i.e. when all the training data resides with a single data owner, and no noise is added to the model coefficients, i.e. ϵ =INF. The other columns correspond to the federated setup from Sec. V with 2 data owners. The results are in line with the observation from Sec. V that the π_{LR} +π_{DP} approach provides higher utility.

TABLE VI
ACCURACY AVERAGED OVER 5-FOLD CV WITH $\Lambda = 1$

DATA SET	n	d	CENTRAL LEARNING 1 DATA OWNER; ϵ = INF	BASELINE (SEC. V-A) 2 DATA OWNERS; $\epsilon = 1$	π_{LR}+π_{DP} (SEC. IV) 2 DATA OWNERS; $\epsilon = 1$
GSE2034	895	12,634	65.55%	51.92%	64.55%
BC-TCGA	1,875	17,814	98.28%	91.37%	95.69%