© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. Published version: https://doi.org/10.1109/ICDIS55630.2022.00036

Epsilon-Differentially Private and Fully Secure Logistic Regression on Vertically Split Data

Angelo Saadeh
Department of Computer Science and Networks
Télécom Paris
Palaiseau, France
angelo.saadeh@telecom-paris.fr

Vaibhavi Kumari
Department of Computer Science
National University of Singapore
Singapore
kvaibhavi@nus.edu.sg

Stéphane Bressan
Department of Computer Science
National University of Singapore
Singapore
steph@nus.edu.sg

***Abstract*—Collaborative networks of organisations need solutions for the secure federated training of machine learning models in distributed environments and for the privacy preservation of the models' parameters after their publication. We propose, present, and evaluate a two-party fully secure logistic regression on a vertically split dataset using an epsilon-differential privacy (DP) mechanism and secure multi-party computation (MPC) protocols. The idea of combining DP and MPC is not new; however, existing solutions leak more information than the one we propose. We empirically evaluate the performance of the proposed mechanism and comparatively argue its originality and relevance in the light of state-of-the-art related work.**

***Index Terms*—Secure multi-party computation, ϵ-differential privacy, Sensitivity, Privacy preserving, Vertical logistic regression**

I. Introduction

Modern organisations train and use machine learning models to support business decisions and to optimise processes. Collaborating organisations, for instance, multiple government agencies, may see the need and the benefits of federatively building such models and, possibly, sharing them with third parties. In this process, the privacy of the data used to train the models needs to be protected against attacks not only from members of both the collaborating organisations among themselves but also from third party users of the models. Indeed, [1] and [2] show that machine learning models can be attacked (inversion attacks) as they may unwantedly reveal information about the training data.

Secure multi-party computation and ϵ-differential privacy devise solutions to address the issues of protection from collaborating organisations and from users of the models separately. Can these be combined to form a unified solution? We hereby propose, present, and evaluate a two-party ϵ-differentially private and fully secure logistic regression on a vertically split dataset. In this process, the players need to jointly train a model, rather than independently train models from their local data as horizontal federated learning players do.

We consider two players, P_1 and P_2, representing two collaborating organisations, who jointly own a vertically split tabular dataset $D \in \mathbb{R}^{r,d+1}$ containing private data with r entries represented in rows, d feature columns, and a label column for the sake of the classification task. P_1 owns D^1 which are d_1 columns of the dataset D, and P_2 owns D^2, d_2 columns of D - including the label column - with $d_1 + d_2 = d$ and $D^1||D^2 = D$. We also consider a user, U, consuming the model. The reader notices that the players, P_1 and P_2, are also consumers of the model. Their common goal is to compute a logistic regression on the whole dataset such that P_i doesn't reveal D^i to the other player, and that the result is ϵ-differentially private. More precisely, their goal is to obtain the model $W^* \in \mathbb{R}^d$ (or weights) produced by a logistic regression performed on the concatenated dataset with respect to the classes defined by the label column. Most importantly, the players want the computation to be private and secure, as follows.

Under ϵ-differential privacy, in the sense of [3], the model obtained should, by itself, give to its consumers only exponentially small information, with respect to ϵ, on whether a particular entry was or was not in the dataset. Furthermore, under security, in the sense that the actual computation process should, by itself, give no more information about each other's columns than what they could have deduced from the final result and their own columns.

In some protocols we consider that players are initialized, in a setup phase, with correlated randomness. This data is independent from their inputs and outputs. Thus it can be for example distributed to them by a trusted third party ahead of the protocol, or generated between them using what is known as a "pre-processing" protocol [4]. For efficiency we chose the first solution, hence the work is implemented with a neutral third party whose sole role is to generate random elements and distribute them to P_1 and P_2, without actively participating in the computation.

We assume in this paper that players follow what is denoted as the "honest-but-curious" model [5]. Namely, players correctly execute the specified computation process. Nonetheless, they may passively try to infer from this computation process as much information as they can. On the other hand, active security cannot be achieved with our proposed solution.

We implement our work in PySyft (version 0.2.9), which is a Python library for secure and private deep learning where real numbers are represented in the ring $\mathbb{Z}/2^l\mathbb{Z}$ with $l \in \{32, 64\}$.

978-1-6654-5968-6/22/$31.00 ©2022 IEEE

The PySyft library includes MPC protocols that we will be using. We use the PySyft framework with two players, P_1, P_2 and a so called "crypto provider" who is the neutral player that doesn't own any data, and whose sole job is to provide P_1 and P_2 with crypto-correlated randomness that they might need during the protocol.

II. Related Work

Our solution works for all dataset separations, but most of the related works focus on horizontal federated learning, which is comparatively easier than for other separations, including the vertical one. Different related works offer different notions of security. For example, [4], [6] and [7] benchmark their framework performing machine learning inference using linear regression, logistic regression, and neural network models. These secure multi-party computation primitives for machine learning do not yet consider differential privacy in their frameworks as we do, and it is important to do so because of the aforementioned inversion attacks. Other existing works like [8] and [9] describe protocols for private federated learning on vertically partitioned data that consider differential privacy, but only (ϵ, δ)-DP - which is a weaker guarantee than ϵ-DP considered in this paper. The ϵ-DP technique that we use is called functional mechanism [10] (FM). It adds noise to the loss function that is being minimized in order to obtain the final weights. The authors of [11] propose a solution that computes an ϵ-DP logistic regression on a vertically split dataset. They use the same DP mechanism as we do, but their algorithm reveals information to a third party, and ours does not. The approach used by [12] also achieves an ϵ-DP logistic regression computed on a vertically split dataset, and like us, they do not reveal any intermediary results. However the differential privacy mechanism that they use is "output perturbation" whose limitations compared to FM - the mechanism that we use - are addressed in [10]. Finally, some works like [13] use homomorphic encryption, which is a form of encryption that allows users to perform computations on encrypted data. However, [14] reports speed performance 100 to 1000 times slower than secure MPC. This is why we did not consider this approach.

III. Background

A. Logistic Regression

Logistic regression [15] estimates the parameters of a logistic function that best classifies the objects of study. After this training phase, the "model", namely the parameters of the logistic function, is shared and used as a classification device. Logistic regression is not only widely used but also is characteristic of the essential mechanisms found in more complex machine learning models such as neural networks. In our work, a logistic regression algorithm minimizes a loss function 1 (also called objective function) by using the gradient descent method.

$$f_D(W) = \sum_{i=1}^{n} \left[\log\left(1 + \exp(x_i^T \cdot W)\right) - y_i x_i^T \cdot W\right]. \quad (1)$$

The obtained minimum W^* of f_D is the model that will be used to classify the data points of D.

B. Secure Multi-party Computation

Secure multi-party computation is a cryptographic tool that allows a set of n players $P_1, ..., P_n$ to collaboratively compute a public function $f(x^1, x^2, ..., x^n)$ where $x^i \in \mathbb{Z}/2^l\mathbb{Z}$ - a finite ring - is the secret input of P_i without having any of the players reveal these secret values. In other words, MPC allows players to make computations on secret values. This work uses MPC to compute the function corresponding to the logistic regression algorithm given the private data owned by two parties P_1 and P_2 as input. Our choice to use MPC protocols instead of multi-key homomorphic encryption is motivated by the fact that MPC performs 100x faster for machine learning tasks.

As in any cryptographic context, an adversary may try to corrupt players to have access to their private information or even sabotage the MPC protocol. The protocols defined in this paper can resist passive corruption - i.e. the players will not deviate from the protocol. In addition, we assume that the channels of communications between the players are secure.

Definition 1 (Secure): Let Π_f be a 2-party protocol that computes a function f. We say that Π_f is secure if the players learn no more information than they could have deduced from the sole final output and their input.

Our protocols are based on the two-party additive secret sharing scheme, which is a method that allows a player P_i to "encrypt" a secret by sending a fragment - called share - of his secret to the other player P_j and himself.

Example 1 (Two-party additive secret sharing): If player P_i with $i \in \{1, 2\}$ wants to share their private input s^i, then P_i selects s_1^i, s_2^i uniformly at random in a finite ring $\mathbb{Z}/2^l\mathbb{Z}$, such that $s_1^i + s_2^i = s^i$, and sends s_j^i to the player P_j for $j \in \{1, 2\}$. We call s_j^i the share of P_j of the secret s^i.

This secret sharing scheme requires the participation of all the players in order to rebuild the secret, and it is information theoretic private. In order to open a secret s, all the players need to add their shares s_j of s together so they can obtain $s = \sum_j s_j$.

Let s be a secret value, we will denote the shares $s_1, s_2 \in \mathbb{Z}/2^l\mathbb{Z}$ by $[s]$, such that $s_1 + s_2 = s$. The secret-sharing protocol is denoted by Π_{share} and it consists of choosing two elements such that their sum is equal to the secret, and sending one of these elements to the other party. The operations in MPC protocols will be executed on the received shares.

The goal of using secure MPC is to obtain $[W^*] : (W_1^*, W_2^*)$ shares of W^* given $[D^1] : (D_1^1, D_2^1)$ and $[D^2] : (D_1^2, D_2^2)$ which are the secret shared parts of the dataset owned by P_1 and P_2 respectively. MPC protocols will be used to compute shares of W^* given shares of D^1 and shares of D^2. Each operation of the algorithm will use a protocol, for example given the shares $[x] : (x_1, x_2)$ and $[y] : (y_1, y_2)$ as input, a secure addition protocol denoted by Π_{add} outputs $[x + y] : (z_1, z_2)$ such that $z_1 + z_2 = x + y$, and a secure multiplication protocol denoted by Π_{mult} outputs $[t] : (t_1, t_2)$ with $xy = t_1 + t_2$ and without revealing information about

x or y. An addition protocol can be executed locally without communications between the players (see [16]). However, a share of the multiplication of two secrets cannot be obtained locally, communications are required between the players. A protocol that outputs shares of the product of two secrets was introduced by D. Beaver [17].

Furthermore, as secure MPC protocols are performed in a finite ring, real decimals will be mapped automatically by the PySyft framework to $\mathbb{Z}/2^l\mathbb{Z}$ (see [18]).

A secure MPC Protocol can be compared to a black-box or an ideal executioner to which players send their secrets and get an output, however sometimes the players can deduce information from the output. No MPC protocol could, in terms of privacy, outperform an ideal executioner of the protocol that computes the function and sends back the results. Thus, the only possible source of weaknesses lies in the specification of the function itself being computed, hence the need for a differentially private method that adds well chosen noise during the computation.

C. Differential Privacy

The privacy of the dataset used to train a logistic regression [15] model using only secure MPC - such that none of the training parties see each other's slice of the dataset - still can be attacked and violated [3] once the parameters of the model are published to the user U. Thus in order to provide a stronger privacy guarantee on the data, we propose using ϵ-differential privacy combined with secure multi-party computation. Reference [3] introduced the notion of ϵ-differential privacy, where the output of an algorithm operating on a dataset does not allow the identification of any individual data point. By incorporating random noise, the algorithm protects the confidentiality of information such that an adversary user can not infer the presence of any data instance in the dataset by observing the model parameters. Furthermore and more specifically, we adopt the functional mechanism (FM) proposed in [10] because of its good privacy guarantee.

In order to define ϵ-differential privacy, we consider the same definition of neighboring datasets as in [3], i.e. two datasets that differ in only one tuple.

Definition 2 (ϵ-Differential Privacy [3]): A randomised algorithm f satisfies ϵ-differential privacy if for any two *neighbouring* datasets D and D' and for any output S of f the following relation satisfies:

$$Pr[f(D) = S] \leq e^{\epsilon} Pr[f(D') = S]. \tag{2}$$

Here ϵ is a privacy parameter where a smaller value of ϵ guarantees more privacy. From 2, if an algorithm f satisfies ϵ-differential privacy, then the probability distribution of f's output S is roughly the same for any two databases that differ in one tuple.

We use the functional mechanism [10] to train the model. In this mechanism, the privacy guarantee is achieved by perturbing the objective function f_D in 1 during the training process by adding to its coefficients Laplacian noise - which correspond to elements sampled obeying a Laplacian distribution $\mathcal{L}$. According to [10], and given

$$\beta = \frac{\Delta}{\epsilon}, \Delta = 3d + \frac{d^2}{4}, \text{ and } \eta_0, \eta_1, \eta_2 \sim \mathcal{L}(0, \beta),$$

the perturbed objective function $\tilde{f}_D(W)$ with the data of x, y is given in 3:

$$\tilde{f}_D(W) = (\log(2)) + \eta_0 + \left(\frac{x}{2} - y * x + \eta_1\right) W + W^T \left(\frac{x^T \cdot x}{8} + \eta_2\right) W, \tag{3}$$

where the operator $*$ denotes an element-wise multiplication. The derivative of the perturbed objective function of logistic regression $\tilde{f}_D$ with respect to the weights, denoted as ∇W, will be used as a perturbed gradient to update the weights during the training process of the algorithm of the logistic regression.

IV. Achieving an Efficient Implementation

There exist MPC protocols for comparisons, and we use a comparison protocol from [19] - denoted by Π_{comp} and referred to as "function secret sharing" (FSS) - that is available in the PySyft 0.2.9 library, it performs a secure comparisons in only one round in the online phase. Let $\tilde{s}, \tilde{a} \in \mathbb{R}$ such that their representation in the finite ring is $s, a \in \mathbb{Z}/2^l\mathbb{Z}$, with $[s] : (s_1, s_2)$ and a public. The comparison operator will be computed using Π_{comp} which outputs $[\tilde{s} < \tilde{a}] : (\beta_1, \beta_2)$.

The steps of adapting the logistic regression algorithm to secure multi-party computation are the following. First, the data normalisation is done locally by P_1 and P_2. As the data is split vertically, the normalization part does not need information from the columns of the other party. Second, real decimal values are mapped to $\mathbb{Z}/2^l\mathbb{Z}$. This is done automatically in PySyft. Third, secret sharing, addition and multiplication protocols are already implemented in PySyft, and they are used to compute the regression. Finally, the accuracy is computed using the appropriate protocol and the players open the weights W^* by adding their shares together. The operations in the logistic regression are computed by evaluating consecutively addition, multiplication and comparison MPC protocols.

A. Sigmoid Function in MPC

Computing a logistic regression requires at some point computing the function sigmoid:

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{4}$$

This function contains an exponentiation and a division which cannot be computed using MPC. The idea is to express the sigmoid function with additions, multiplications, and comparisons. We use an MPC-friendly approximation of the sigmoid function, that we denote S and define below, which is linear in

three pieces. A similar approximation was introduced in [20]. For $x \in \mathbb{R}$:

$$S(x) = \begin{cases} 0 & \text{if } x < -2 \\ \frac{x}{4} + \frac{1}{2} & \text{if } -2 \leq x \leq 2 \\ 1 & \text{if } x > 2. \end{cases} \quad (5)$$

Our reason for using this approximation is for increased performance. Indeed, the protocol Π_{comp} from [19] allows to construct a protocol Π_{sigmoid} that outputs the result of the function S defined in 5 as the following. Given $\tilde{x} \in \mathbb{R}$ and its representation in the finite ring $[x] : (x_1, x_2)$, Π_{sigmoid} securely outputs $[S(\tilde{x})] : (y_1, y_2)$ by linearizing the function S:

$$S(x) = (1 - (x < -2)) \left(\left(\frac{x}{4} - \frac{1}{2} \right) (x < 2) + 1 \right). \quad (6)$$

As a result, the overall MPC computation of S, denoted as Π_{sigmoid} uses a total of only 4 rounds in the online phase - where the computation protocols take place, as opposed to the offline phase. The results demonstrate that the accuracy of the model trained with MPC and the approximation S of sigmoid is similar to the accuracy of the model trained on data in clear with the actual sigmoid function.

B. Generation of Random Noise

In order to achieve differential privacy, the Laplace noise used in the gradient descent 3 should be unknown to the user of the model U, but also to P_1, P_2 as they are potential users of the model. Therefore, we present a protocol that generates shares of Laplace noise for P_1 and P_2, still without them knowing the value of this noise. We use a standard technique in which we proceed in two steps. The first step is a protocol that generates shares that correspond to a uniformly random element in $(0, 1]$ unknown to the parties. The second step obtains a shared random element following a Laplace distribution. More precisely, computing $[u] : (u_1, u_2)$ where $u \sim \mathcal{U}(0, 1)$ is done with the protocol denoted as Π_{unif} in which P_1 and P_2 generate two elements $a, b \sim \mathcal{U}(0, 1)$ and secret-share them. Then the players compute shares of u using Π_{comp} and 7:

$$u = \begin{cases} a + b & \text{if } a + b < 1 \\ a + b - 1 & \text{if not.} \end{cases} \quad (7)$$

Our implementation of the protocol Π_{unif} takes one round of communication, since we use FSS [19] (available in the PySyft library) for the comparison sub-protocol. Subsequently, given $[u]$ the next step is to apply the protocol denoted as Π_{Lap} to compute shares of the Laplacian noise $[F^{-1}(u)]$ where:

$$l = F^{-1}(u) = -\beta \operatorname{sign}(u - \frac{1}{2}) \log(1 - 2|u - \frac{1}{2}|). \quad (8)$$

This requires computing the natural logarithm, and in order to do so we use the natural logarithm sub-protocol $\Pi_{\log}$ that is already implemented in the PySyft library, and that given $[x]$ securely outputs $[\log(x)]$. Obtaining $l \sim \mathcal{L}(0, \beta)$ using $u \sim \mathcal{U}(0, 1)$ and the function F^{-1} is known technique [21] used to compute random elements following a Laplace distribution in an MPC manner.

For convenience, we directly used the implementation of logarithm in PySft. Notice that an alternative implementation of the natural logarithm is presented in [21], also to generate Laplace noise in MPC. Also, in our new protocol Π_{unif} for uniform random generation in $[0, 1)$, we do not exclude that other protocols like [22] may be more efficient in terms of computation in the online phase. However, our approach nevertheless reaches an optimum in terms of communication latency since it is in a single round.

Algorithm 1: Protocol Π_{train}

Input: $[D]$ the secret-shared dataset,
$[W]$ the secret shared weights initialized using a normal distribution,
ϵ the privacy budget,
α the learning rate,
$epochs$ the number of epochs,
r the number of rows in the training dataset.
Output: The players hold $[W^*]$.

Normalize data locally
Split data into training and testing data
Secret-share data using Π_{share}
$\beta \leftarrow \frac{\Delta}{\epsilon} \cdot epochs$
for *every epoch* **do**
  $[\eta_1] \leftarrow \mathcal{L}(0, \beta)$ using Π_{Lap}
  $[\eta_2] \leftarrow \mathcal{L}(0, \beta)$ using Π_{Lap}
  $[x] \leftarrow$ concatenated feature columns of the training data from $[D]$
  $[y] \leftarrow$ the label column of the training data from $[D]$
  $[\nabla W] \leftarrow [\left[x^T (1/2 - y) + \eta_1\right] + 2 \left[\frac{x^T x}{8} + \eta_2\right] W]$
  $[W] \leftarrow [W - \alpha \cdot \frac{\nabla W}{r}]$
end
$[W^*] \leftarrow [W]$
return $[W^*]$

For convenience, we directly used the implementation of logarithm in PySft. Notice that an alternative implementation of the natural logarithm is presented in [21], also to generate Laplace noise in MPC. Also, in our new protocol Π_{unif} for uniform random generation in $[0, 1)$, we do not exclude that other protocols like [22] may be more efficient in terms of computation in the online phase. However, our approach nevertheless reaches an optimum in terms of communication latency since it is in a single round.

C. Training Algorithm in a Multi-party Setting

Using the protocols that were defined in the previous sections, a training algorithm that uses the functional mechanism and private data can be defined. We present it in Algorithm 1. The final result - the optimized weights W^* - can be left in a shared form, and used for multi-party predictions.

Theorem 1: Algorithm 1 computes a logistic regression that satisfies ϵ-differential privacy as per definition 2 and security as per definition 1.

To compute the accuracy of the model W^* without revealing their testing data, the players can use a protocol denoted as Π_{acc} that securely returns the accuracy γ of the model using the testing data x_{test}, y_{test}, and the optimized weights W^*.

$$\gamma = 1 - \text{mean}(\text{round}(|\operatorname{sigmoid}(x_{test} \cdot W^*) - y_{test}|)). \quad (9)$$

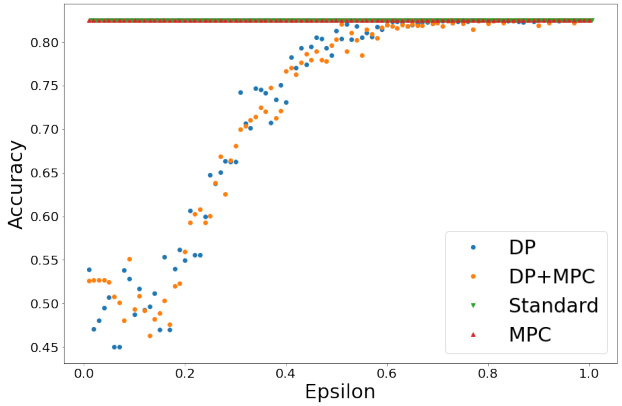


Fig. 1. Accuracy of models trained on the Brazil dataset

This protocol denoted as Π_{acc} uses the protocol Π_{sigmoid} to compute the sigmoid function and the comparison protocol Π_{comp} to perform the absolute value and rounding operations. These operations are computed using the following linearized equations, given x, and $y \in [0, 1]$:

$$\text{abs}(x) = 2x(x > 0) - x \text{ and } \text{round}(y) = (0.5 > y).$$

V. Experimental Results

We perform two kinds of evaluation to check the effectiveness of our approach. First, we compare the accuracy of the following four logistic regression models: the original logistic regression, referred to as "Standard", our secure multi-party logistic regression, referred to as the "MPC". "Standard" and "MPC" are independent from the parameter ϵ. Our implementation of a functional mechanism-based ϵ-differential private logistic regression is referred to as "DP", and our proposed secure multi-party functional mechanism-based ϵ-differential private logistic regression is referred to as "DP+MPC".

We use the Brazil census dataset from IPUMS and the Adult dataset for our experiments. For each dataset, we normalise each feature line to ensure that it lies within the range of [-1, 1]. We also normalize the samples in each dataset such that $||x_i||_2 \leq 1$ for $i \in 1, ..., r$. After the cleaning and preprocessing, we chose a subset of the Brazil dataset with 14,000 rows and five columns. After normalising the numerical features and encoding all the relevant categorical features of the Adult dataset, we have a dataset with 18,000 rows and eight columns.

The accuracy γ is measured as the average number of times the model is able to assign correct values to the target column for a new data point. For both datasets, the regression is run 100 times, for ten epochs each time, with all the data in one batch.

The figures compare the accuracy achieved by the four models mentioned above. It shows the accuracy of the models on varying the value of ϵ from 0.01 to 1. Naturally, the accuracy of the original and MPC model does not depend on the perturbation introduced by the differential privacy mechanism as seen in Fig.1 and Fig.3. The ϵ-differential

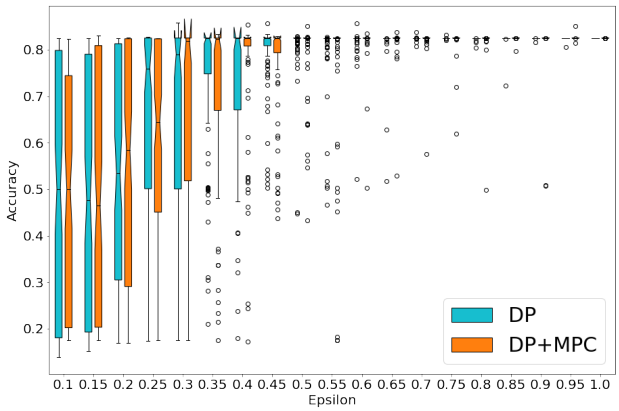


Fig. 2. Variation of accuracy on the Brazil dataset

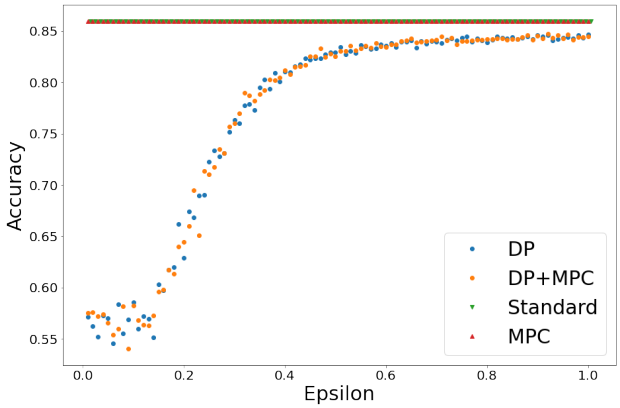


Fig. 3. Accuracy of models trained on the Adult dataset

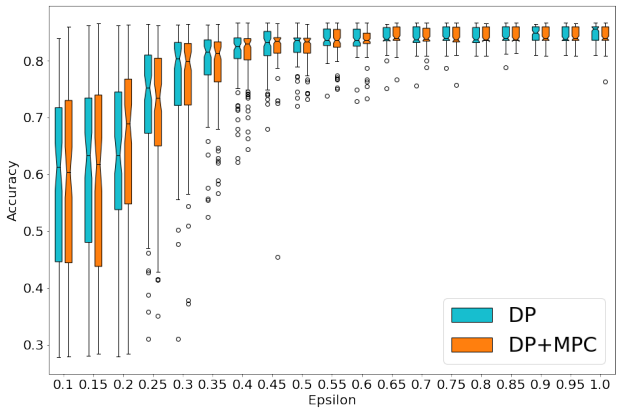


Fig. 4. Variation of accuracy on the Adult dataset

private mechanism introduces a trade-off between the privacy parameter ϵ and the accuracy for ϵ-DP and ϵ-DP MPC model. The accuracy of both these models improves as the value of ϵ increases. This is in accordance with the theory that a stronger privacy guarantee results from extra noise that deteriorates the utility. Fig.2 and Fig.4 give a boxplot of the accuracy achieved by the ϵ-diff model and the ϵ-diff MPC model for 100 regression runs. Small values of ϵ show significant variance in the accuracy of the models. As the privacy parameter is relaxed, the variance in the accuracy becomes less, and the median accuracy becomes closer to the actual accuracy of the model.

The goal was to add MPC to the DP model without impacting the accuracy, and it is achieved. On combining the secure multi-party computation model and ϵ-differential privacy into the mechanism that we propose in this paper, we see that we can achieve a fully secure and privacy-preserving mechanism at a good economy of accuracy.

VI. Conclusion and Future Works

We have proposed, presented, and evaluated an original ϵ-differentially private and fully secure logistic regression on a vertically split dataset. As explained, secure multi-party computation ensures security when performing computations. However, it does not guarantee user data privacy if the output of the computation is made public. Adding a layer of differential privacy over secure multi-party computation guarantees additional data privacy.

The solution proposed utilises the functional ϵ-differential privacy mechanism and a fully secure original multi-party stochastic gradient descent computation for the logistic regression.

The functional mechanism for differential privacy provides fine-grain control over privacy and accuracy. We have used secure multi-party computation without homomorphic encryption for the advantages that secure multi-party computation offers in terms of efficiency. While functional mechanism guarantees the model's privacy by adding noise to the objective function, which affects the model's accuracy, secure MPC is a tool that will only compute the desired function without seeing the input, which does not affect the result of the computation.

Our empirical studies on the dataset show that the impact of the addition of secure multi-party computation over differential privacy is very close to training using the concepts of differential privacy only. The evaluation of performance and the comparative discussion of the proposed mechanism show that it constitutes an original effective, efficient, secure, and privacy-preserving solution.

We chose to compute a logistic regression because it constitutes a fundamental building block for more complex machine learning models. We are now considering the transposition of the original contributions of this paper to privacy-preserving and secure multi-layer perceptrons and neural networks.

Acknowledgement

This research is partially supported by the National Research Foundation, Prime Minister's Office, Singapore, under its Campus for Research Excellence and Technological Enterprise (CREATE) programme as part of the programme DesCartes.

References

[1] 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*, ACM, 2015.

[2] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2017.

[3] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, 2014.

[4] I. Damgård, D. Escudero, T. K. Frederiksen, M. Keller, P. Scholl, and N. Volgushev, "New primitives for actively-secure MPC over rings with applications to private machine learning," in *2019 IEEE Symposium on Security and Privacy*, IEEE, 2019.

[5] R. Cramer, I. Damgård, and J. B. Nielsen, *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[6] P. Mohassel and P. Rindal, "Aby^3: A mixed protocol framework for machine learning," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS*, ACM, 2018.

[7] D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl, "Improved primitives for MPC over mixed arithmetic-binary circuits," in *Advances in Cryptology - CRYPTO 2020*, vol. 12171, Springer, 2020.

[8] R. Tajeddine, J. Jälkö, S. Kaski, and A. Honkela, "Privacy-preserving data sharing on vertically partitioned data," *CoRR*, vol. abs/2010.09293, 2020.

[9] C. Wang, J. Liang, M. Huang, B. Bai, K. Bai, and H. Li, "Hybrid differentially private federated learning on vertically partitioned data," *CoRR*, vol. abs/2009.02763, 2020.

[10] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett, "Functional mechanism: Regression analysis under differential privacy," *CoRR*, vol. abs/1208.0219, 2012.

[11] D. Xu, S. Yuan, and X. Wu, "Achieving differential privacy in vertically partitioned multiparty learning," *CoRR*, vol. abs/1911.04587, 2019.

[12] S. Pentyala, D. Railsback, R. Maia, R. Dowsley, D. Melanson, A. C. A. Nascimento, and M. D. Cock, "Training differentially private models with secure multiparty computation," *CoRR*, vol. abs/2202.02625, 2022.

[13] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang, "Fast and secure linear regression and biometric authentication with security update," *IACR Cryptol. ePrint Arch.*, 2015.

[14] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS*, ACM, 2019.

[15] J. Berkson, "Application of the logistic function to bio-assay," *Journal of the American statistical association*, vol. 39, 1944.

[16] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract)," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, ACM, 1988.

[17] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology*, vol. 576, Springer, 1991.

[18] O. Catrina and A. Saxena, "Secure computation with fixed-point numbers," in *Financial Cryptography and Data Security, 14th International Conference, FC 2010*, vol. 6052, Springer, 2010.

[19] T. Ryffel, P. Tholoniat, D. Pointcheval, and F. R. Bach, "Ariann: Low-interaction privacy-preserving deep learning via function secret sharing," *Proc. Priv. Enhancing Technol.*, 2022.

[20] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy*, IEEE Computer Society, 2017.

[21] K. Thissen, I. L. Schoenmakers, I. R. Koster, and I. P. van Liesdonk, *Achieving Differential Privacy in Secure Multiparty Computation*. PhD thesis, Eindhoven University of Technology, 2019.

[22] G. Wu, Y. He, J. Wu, and X. Xia, "Inherit differential privacy in distributed setting: Multiparty randomized function computation," *CoRR*, vol. abs/1604.03001, 2016.