Enhanced Privacy-Preserving Decision Trees using Secure Multiparty Computation and Differential Privacy

Anonymous Author(s)

ABSTRACT

We address the problem of decision tree learning from data that may be distributed across multiple data owners while protecting the privacy of the training data and the model from each data owner. To protect the training data, Secure Multiparty Computation (MPC) provides a very promising privacy-preserving solution with none or minimum accuracy loss. However, there is a potential privacy leakage on the training data during model prediction or inference. To this end, existing solutions use a hybrid framework that combines MPC and Differential Privacy (DP), where DP models are trained with MPC protocols. The existing approaches combine MPC and DP in a naive way and often lead to models with lower accuracy. In this work, to take the full advantage of MPC's inherent security guarantee, we propose a novel way of utilizing both MPC and DP that can improve model accuracy while providing the same privacy guarantee. Our key design idea is to adopt MPC for building the entire model without leaking any intermediate results, and then DP noise is only added at the leaf level to achieve the desired ϵ-DP. By doing so, less amount of noise is needed, and as a consequence, model accuracy can be significantly improved. We provide formal security proof of the proposed protocol and analyze the amount of required DP noise. In addition, we implemented our protocol in a distributed environment, and our empirical results show that our approach can indeed improve model accuracy by up to 29% for the Adult dataset even with a small privacy budget of $\epsilon = 0.005$ comparing to the existing solution. On the other hand, our solution is computationally more expensive, and it trades off between accuracy and computation cost.

ACM Reference Format:
Anonymous Author(s). 2024. Enhanced Privacy-Preserving Decision Trees using Secure Multiparty Computation and Differential Privacy. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Training machine learning (ML) models on distributed data while providing formal privacy guarantee regarding the training data brings a great value in various business settings. Without a privacy protection mechanism in place, data scientists may not be able to build any ML models. Suppose that data scientists want to build ML models on customer data from multiple organizations. Under the traditional and centralized solution, customer data need to be collected and managed by a central server. Data scientists interact with the server and perform the required computations to generate ML models. This centralized solution may not be always possible due to privacy risks and government regulations, such as the General Data Protection Regulation (GDPR) [29].

To address potential privacy risks, privacy-enhancing technologies have emerged as promising solutions that can help data scientists analyze customers' data through the ML life-cycle without revealing private information in a distributed/federated environment. The common techniques include Federated Learning (FL) [21, 25, 42], Homomorphic Encryption (HE) [5, 14], Secure Multiparty Computation (MPC) [43] and Differential Privacy (DP) [9], where each of these techniques has pros and cons. In FL, while data never leave the data owners, the resulting model can be less accurate comparing to the centralized solution. Additionally, intermediate model parameters may leak information about each party's local training data. Recent works have studied secure training of decision trees under federated learning with HE and/or MPC. These techniques rely on cryptographic schemes and allow multiple parties to jointly train decision trees without revealing each party's private data. Also, they can provide as good accuracy as the centralized solution. However, the output model may not necessarily be private and does not provide a formal DP guarantee. It is well known that models are vulnerable to membership inference attack [27, 36]. In other words, when models and/or prediction results are shared with data scientists or a third party, they can leak private information about individuals in the training data.

Independently, DP decision trees have extensively studied in the context of centralized learning [24] where decision trees are learned over training data by answering queries under DP. A resulting model that satisfies DP can be publicly released for inference without degrading privacy loss. Because noise is introduced during a model training to satisfy DP, there is a cost of accuracy loss. The accuracy is mainly affected by the total privacy budget, the number of queries required to build decision trees, and the sensitivity of the queries. Since the server, which produces the DP-trees, has access to the original data, the scheme does not work in a federated environment where the participating parties do not want to disclose their private training data.

In order to protect the training data during both training and inference, a hybrid framework was introduced in [40] that combines MPC and DP by training a centralized DP decision tree using MPC. However, the model accuracy provided by this approach is limited to what can be achieved by the centralized DP decision tree solutions [24, 40]. In addition, as far as we know, there is no implementation and empirical evaluation of the unified framework of MPC and DP for decision tree training. Thus, the overhead of adding DP into MPC (or vice versa) during training is an open question. Furthermore, a synergy between MPC and DP is not well understood as the two

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

techniques are often considered as orthogonal problems. In fact, we provide a novel way of combining MPC and DP that can provide a better accuracy than the existing approach under the same DP guarantee while offering data privacy and model confidentiality. Overall, the existing solutions do not satisfy one or more of the following properties:

- Data privacy: Each party's private training data should not be disclosed during model training and inference.
- Accuracy: The model should be as accurate as possible.
- Model confidentiality: This property is essential when the model has monetary value and disclosing the model may open the door for more effective inference attacks.
- Query privacy: Query content and prediction result should be hidden from the ML service providers since query content may contain sensitive information.

1.1 Problem Definition

We consider a cloud-assisted setting where data scientists build tree-based models using cloud computing nodes for clients who own private data. Prediction results are shared with data scientists. Our goal is to train a DP tree-based model as if the model were trained on a trusted server without participating parties sharing their private training data. The proposed protocol achieves all four properties defined previously. In order to maximize efficiency, we utilize a multi-server model where data owners/clients secretly share their training data with three independent computing servers. Model training is performed by the servers using secret shares, and the learned model is also secretly shared and stored at the servers. A user or data scientist secretly shares its query with the servers who perform the required computation to derive the prediction result in secret share format. The shares are sent to the user who then reconstructs the actual prediction result. To summarize, we consider the following three classes of entities:

- Clients C: Each client C_i holds private training data and secretly shares its data with the cloud servers.
- Servers S: The computing servers manage secretly shared data from the clients and collaboratively perform MPC-based training protocols. The servers also store the trained model and provide inference service to data scientists/users.
- Users U: A user issues a query to the servers and receives shares of the prediction result. A user may be affiliated with a client or is authorized to the inference service.

1.1.1 Data Partitioning. The proposed solution works for one or more clients. When there are multiple data owners, we assume that the data are either horizontally or vertically partitioned.

- Horizontal: the training data from each client have the same features but correspond to a disjoint-set of individuals.
- Vertical: the training data from each client have a disjoint-set of features but correspond to the same individuals.

The multi-server setting offers a high-level scalability especially when a number of clients need to build an ML model on their aggregate data. Moreover, it works for either horizontal or vertical data partitioning scheme.

1.1.2 Threat Model. We consider the semi-honest model [15] where an adversary follows the protocol but may try to infer the other parties' private information based on its own input, output and the messages received during protocol execution. We also assume that the participating parties do not collude. On a high level, here we briefly emphasize how data privacy is achieved from the perspective of each entity group:

- Clients: Each client secretly shares its data with the servers and does not participate in any other computations related to model training and inference. Thus, it cannot learn any information about the other clients' training data.
- Servers: They only perform computations on secretly shared data, and the original data are never reconstructed at the servers. In addition, any intermediate results, accessible to the servers, are either secretly shared or randomized. At a consequence, the servers cannot learn any information about the training data, except for certain domain knowledge, e.g., the domain size of each attribute.
- Users: Because the prediction result is differentially private, the user cannot learn anything about the training data more than what DP protects.

1.2 Our Contribution

Adopting the multi-server setting, we design and implement MPC protocols for training tree-based ML models. The proposed protocols satisfy: data privacy, high model accuracy, and model confidentiality, and query privacy. Our solution work for any number of clients and users, but it requires at least three independent computing servers to perform secretly shared based MPC computations. Although we assume the semi-honest adversary model, our solutions can be easily extended to satisfy the malicious model. In summary, this work has several significant contributions:

- We propose a novel way of combining MPC and DP for tree based ML models that requires less amount of noise and with provable DP guarantee.
- A formal security proof is provided to theoretically justify that our approach can achieve the same ϵ-DP by only adding noise at the leaf level.
- Inspired by the high level discussions given in [8], we develop concrete and fully secure ID3 tree generation and prediction protocols.
- The proposed protocols are implemented in a real distributed environment to generate extensive empirical data. The results are consistent with our design and theoretical analysis.

The rest of the paper is organized as follows: Section 2 presents the existing solutions closely related to our work, Section 3 details our proposed protocol, Section 4 conducts extensive experiments using three independent servers to study protocol scalability and model accuracy, and Section 5 concludes the paper with lesson learned and future research directions. In the appendix, we provide technical background on DP, implementation details of the existing solution, additional details and security analysis of the proposed protocol, and an extension to random forest.

2 RELATED WORK

Differentially private decision trees have been well studied in the centralized setting, including decision trees, random forests and XGBoost [4, 7, 11–13, 18, 23, 41]. However, in the centralized setting, a party who builds a model is assumed to have access to the entire training dataset. Orthogonally, MPC-based decision tree training has been studied where a model is learned on secret-shared data [1, 8, 16]. Hoogh et al. proposed MPC-based protocols for an ID3-based tree algorithm for cases of revealing and hiding the tree [8]. Recent works have focused on a secure CART-based tree algorithm while keeping the model secret [1, 16]. While the training data are protected, those protocols do not provide DP guarantees and thus are susceptible to membership inference attacks.

The combination of MPC and DP for tree-based models is considered in several works, especially in the context of federated learning where DP is involved in various contexts. One common usage of DP is to replace cryptographic techniques for efficiency gain. This approach is computationally efficient, but it incurs a significant loss in utility due to the added noise to achieve local DP [22, 38].

Another usage is to train DP models in the federated setting to make them robust to membership inference attacks, which align with our motivation [24, 39, 40]. Maddock et al. provided a general framework of learning DP gradient boosted decision tree models in a federated setting [24]. Their framework uses secure aggregation-based MPC and DP. However, since the majority operations are performed locally at each party, DP techniques that require centralized evaluation, e.g., the Exponential mechanism, were not considered. This can be a limitation since many DP decision tree-based algorithms require the Exponential mechanism to provide a good accuracy. Our framework relies on a secret-sharing scheme and thus can completely simulate a centralized DP learning in a federated setting. Wu et al. briefly introduced how to train a centralized DP decision tree with an MPC framework in a federated setting [40]. Their approach of combining MPC and DP during tree generation process is very different from ours (adding noise at the leaf level and producing more accurate results), and it only works for vertically partitioned situation. Furthermore, they do not have an implementation and empirical evaluation of the hybrid framework of MPC and DP and thus efficiency of the framework was not clear.

Another essential and often overlooked aspect is that when analyzing DP guarantee, the existing work on DP-tree models neglects the fact that the tree structure itself can reveal information about the training data, and their DP analyses ignore the impact of the tree structure. Therefore, the existing solutions may not achieve the expected DP guarantee. As far as we know, there is no existing work that comprehensively analyzes the connection between MPC and DP for decision trees from the perspective of privacy, accuracy and efficiency. This is because MPC and DP are often considered orthogonal problems. However, we will show that the existing way of combining MPC and DP can result in sub-optimal performance. In addition, by taking advantage of the inherent privacy guarantee of an MPC protocol, we will theoretically demonstrate that adding noise to the leaf nodes (and less noise overall) is sufficient to achieve the same level of DP while simultaneously improving accuracy comparing to the existing approaches.

3 THE ENHANCED MPC+DP FRAMEWORK

In this section, we design an alternative way to implement a differentially private and fully secure ID3 protocol. A more detailed discussion on the existing MPC+DP solutions for generating ID3 tree models is provided in Appendix B. As stated before, the models produced by the existing solutions are not very accurate for smaller ϵ. In addition, the tree structure is leaked which can affect the actual DP guaranteed. Unlike the existing solutions, our approach takes advantage of MPC's inherent 0-DP guarantee when the intermediate computations are not disclosed or remain as secret shares. In other words, if we do not disclose the chosen attribute for the internal nodes, DP-noises are not needed to randomize the attribute selection process. In our formal security analysis given in Appendix C.2, we will prove the 0-DP property in our particular way of using MPC in the tree building process.

3.1 The Proposed Secure ID3 Protocol

To completely hide all intermediate information about the data and keep the tree secret, every non-leaf node must have the same number of branches. This can be achieved by adding dummy values to each attribute so that every attribute $A_i \in \mathcal{A}$ has the same domain size of $w = \max_i |dom(A_i)|$. In other words, we will build a complete w-ary tree except for leaf nodes with $|dom(A_0)|$ labels where A_0 denotes the class attribute. Also, the stopping criterion only checks if the tree reaches the maximum depth.

3.1.1 Data Pre-processing. Before executing the model generation protocol, each data owner needs to locally pre-process its data. We assume that data owners apply consistent pre-processing strategies, such at discretization, handling missing values, etc. In addition, data integration at the servers can be tricky. Under the vertical partition scheme, the data owners generate one hot-encodings for their attributes and then secretly share the encodings with the three computing servers. The servers need to store the shares in the same ordering which can be guaranteed easily even if they do not know the data schema and attribute domains. Under horizontal partition, it becomes more challenging to integrate the shares depending whether or not the schema is known. See Appendix C.1 for a more concrete discussion and a one-hot encoding integration protocol provided by Algorithm 11.

3.1.2 The Tree Generation Protocol. We adopt the following notation conventions:

- $[x]$: secret shares of x, and we use bold lower case letter to represent a vector: $[\boldsymbol{x}_i]$ represents the shares of its i-th component of vector $\boldsymbol{x}$. We also use capital letter to represent a multi-dimensional data or a vector.
- D: the dataset represented as a set of one-hot encodings. D is three-dimensional: $D_{i,j,k}$ represents the k-th component of the j-th one-hot encoding of the i-th attribute. In other words, D_i is matrix and a collection of one-hot encodings for attribute A_i.
- T: a binary vector representing if a tuple is in a dateset.
- R: a set of attribute indices (excluding class attribute index 0) and $\mathcal{R}$ a binary vector of size $|R|$ indicating the available attributes for the current recursive call.
- h: the tree depth is bounded by $1 \leq h \leq |\mathcal{A} - \{A_0\}|$.
- $[\boldsymbol{x}] \bullet [\boldsymbol{y}]$: represent a secure dot product between two secretly shared vectors.
- $[\boldsymbol{x}] \times [\boldsymbol{y}]$: represent secure component-wise multiplications between two secretly shared vectors.

The key steps of our tree generation protocol (Algorithm 1) include:

- **Compute the class partitions and frequencies** (Steps 5-8): derive a binary vector C_i for each class i. $C_{i,j} = 1$ implies the j-th tuple belongs to class i, and 0 otherwise. These vectors are used to derive class frequencies which are stored in $\boldsymbol{\tau}$. Each frequency is DP randomized using the MPC-LM protocol (Algorithm 12 in Appendix C.3.1).
- **Check the stopping condition and derive the majority class** (Steps 9-11): When h reaches 0, the protocol computes the majority class returned as the leaf label.
- **Compute the quality scores** (Steps 12-14): the C_i vectors derived from the previous steps are used by the MAX protocol (Algorithm 2). At step 14, the scores for the available attributes are kept unchanged, but the scores for the other attributes are set to 0 so that these attributes would not be chosen for the current node.
- **Select the best attribute and its encoding** (Steps 15-19): the arg max protocol returns the index of an attribute whose quality score is the maximum. At step 16, the index is converted into a vector representation using the Indicator protocol (see Algorithm 3 for detail). Then one hot encodings of A_k are retrieved and stored in X.
- **Partition the data and recursively build sub-trees** (Steps 20-22): At step 21, $[T] \times [X_i]$ produces a binary vector that represents the data partitioned based on $A_{k,i}$. $[\mathcal{R}] - [\boldsymbol{k}]$ means that the chosen attributed is removed from further consideration. Then a sub-tree is built with each data partition with tree height reduced by one.

As stated in Appendix C.1, the dummy values are represented as zero vectors. As a result, they have no effect on the MAX protocol, and the quality scores for all attributes are unaffected by the dummy values. Although the resulting tree model has branches generated from these dummy values, they do not interfere with or affect the result of model inference tasks according to our prediction protocol. Section 3.2 provides more details.

Quality metric. There are various quality metrics that determine the best split. While popular metrics are information gain or Gini index, some works explore the median or the max operator. In the context of DP, since the choice of quality metrics affects the model accuracy, the max operator, which has a lower sensitivity, has shown to have a better performance than the other metrics [13]. For this reason, we consider the max operator as the quality metric for choosing the best split attribute. The max operator corresponds to the misclassification rate by picking the class with the highest frequency, which has the sensitivity of 1:

$$q(D, A_i) = \sum_{j \in A_i} max_{c \in A_0} \left(D_{A_{i,j}} \bullet D_{A_{0,c}} \right) \tag{1}$$

The MAX protocol presented in Algorithm 2 is our MPC implementation of the max operator according to Equation 1.

3.1.3 Complexity Analysis. For MPC protocols, the asymptotic complexity is often based on the number of secure multiplications (SM). Since secret sharing based secure addition does not require any communication, its complexity is negligible comparing to SM. First, we analyze the complexity for each sub-protocol, and let l represent the share size in bits:

- $\arg\max([\boldsymbol{x}])$: this protocol can be implemented using $|[x]|-1$ secure comparisons (SC), each of which has a complexity of $O(l)$. Thus, the complexity for arg max is $O(l|[x]|)$.
- $\mathrm{MAX}([D_i], [C], u, w) \rightarrow [\boldsymbol{\gamma}_i]$: since C_i has a size of n, each secure dot product requires n SMs. There are u dot products. The max protocol (step 2 of Algorithm 2) can be implemented using $u-1$ SCs. As a result, the total complexity for the MAX protocol is $O((n+l)wu)$.
- $\mathrm{Indicator}(m, [k]) \rightarrow [\boldsymbol{k}]$: the protocol requires m SCs, so it complexity is bounded by $O(lm)$.

Next we estimate the complexity of each major component:

- Steps 5-8: there are u component-wise secure multiplications with size n each, so that complexity is $O(nu)$.
- Steps 9-11: the size of $\boldsymbol{\tau}$ is u; thus, the complexity is bounded by $O(lu)$.
- Steps 12-13: since MAX is called m times, the complexity is $O(m(n+l)wu)$.
- Step 14: requires $O(m)$ SM operations.
- Step 15: the size of $\boldsymbol{\gamma}$ is m; thus, the complexity is bounded by $O(lm)$.
- Step 16: the complexity is $O(lm)$.
- Steps 17-19: there are wn secure dot products of size m, so the complexity is bounded by $O(mnw)$.
- Steps 20-22: there are w recursive calls, and before each call, a secure component-wise vector multiplication is performed. As a result, the complexity is bounded by $O(nw)$ before performing the recursive calls.

It is clear that the complexities at steps 12-13 dominate the rest of the protocol before each recursive call. In addition, since the vector sizes do not change for all recursive calls, the total complexity is bounded by: $O\left(w^{h-1}m(n+l)wu\right)$, for $1 \leq h \leq m$.

3.1.4 Security Analysis. Under the semi-honest model, it is easy to prove the proposed protocol protects the training data since all operations are performed over secret shares and no intermediate results are ever disclosed. Nevertheless, it is challenging to prove the DP guarantee. Due to space limitations, our detailed privacy/security analysis is provided in Appendix C.2. In the proof, we have shown why it is sufficient to add less noise at the leaf level which provides a theoretic justification on the effectiveness of the proposed approach.

3.2 The Prediction Protocol

Let $\mathcal{A} = A_0, A_1, A_2, A_3, A_4, A_5$ where A_0 is the class attribute. Ignoring the class attribute, the rest can be represented as indicator vectors $\{10000, 01000, 00100, 00010, 00001\}$ from A_1 to A_5 respectively. Suppose that each attribute A_i has three values represented by $\{100, 010, 001\}$, corresponding to $\{a_{i1}, a_{i2}, a_{i3}\}$ respectively or the tree branches from left to right. Let $\Gamma = \{\Gamma_1, \Gamma_2\} = \{\Gamma_1, \{\Gamma_{21}, \Gamma_{22}, \Gamma_{23}\}\}$ represent a decision tree where Γ_1 is the root attribute and Γ_2 is a set of children nodes of Γ_1. If Γ_1 is a leaf node, then it is set to c, one of the class labels; otherwise, it is defined recursively. As an example, a partial tree is given in Figure 1. The complete tree is full 3-nary tree; that is, each internal node has

Algorithm 1 ODP+FMPC-ID3($[D], [T], [\mathcal{R}], R, h, \epsilon$) $\rightarrow \{[\boldsymbol{k}], \{[tr_1], \ldots, [tr_w]\}\}$

$u \leftarrow |[D_0]|$ ▷ the number of classes
$m \leftarrow |R|$ ▷ the number of attributes
$n \leftarrow |[D_{0,0}]|$ ▷ the number of tuples
$w \leftarrow |[D_1]|$ ▷ the domain size of each attribute
for $i \leftarrow 1$ to u **do**
  $[C_i] \leftarrow [D_{0,i}] \times [T]$ ▷ tuples belong to class i
  $[\boldsymbol{\tau}_i] \leftarrow \sum_{j=1}^{n} [C_{i,j}]$ ▷ the size of class i
  $[\boldsymbol{\tau}_i] \leftarrow$ MPC-LM($[\boldsymbol{\tau}_i], \epsilon, 1$) ▷ DP class sizes
if $h = 0$ **then**
  $[c_{i*}] \leftarrow \arg\max([\boldsymbol{\tau}])$ ▷ the class with the largest size
  return $\{[c_{i*}]\}$ ▷ the leaf label
for $i \leftarrow 1$ to m **do**
  $[\boldsymbol{\gamma}_i] \leftarrow$ MAX($[D_i], [C], u, w$) ▷ the i-th attribute's score
$[\boldsymbol{\gamma}] \leftarrow [\boldsymbol{\gamma}] \times [\mathcal{R}]$ ▷ keep the scores of available attributes
$[k] \leftarrow \arg\max([\boldsymbol{\gamma}])$ ▷ the attribute index with the max score
$[\boldsymbol{k}] \leftarrow$ Indicator($m, [k]$) ▷ converting k into a vector format
for $i \leftarrow 1$ to w **do**
  for $j \leftarrow 1$ to n **do**
    $[X_{i,j}] \leftarrow [D_{1,i,j}, \ldots, D_{m,i,j}] \bullet [\boldsymbol{k}]$ ▷ encoding of A_k
for $i \leftarrow 1$ to w **do**
  $[T_i] \leftarrow [T] \times [X_i]$
  $[tr_i] \leftarrow$ ODP+FMPC-ID3($[D], [T_i], [\mathcal{R}] - [\boldsymbol{k}], R, h-1, \epsilon$)
return $\{[\boldsymbol{k}], \{[tr_1], \ldots, [tr_w]\}\}$

Algorithm 2 MAX($[D_i], [C], u, w$) $\rightarrow [\boldsymbol{\gamma}_i]$

Require: $[D_i]$ is a set of w one-hot encodings of A_i. $[C]$ is a set of binary vectors, and $[C_j]$ indicates if a tuple belongs to class j. w defines the domain size of all attributes, and u indicates the number of classes.
for $j \leftarrow 1$ to w **do**
  $[\boldsymbol{\tau}_j] \leftarrow \max([D_{i,j}] \bullet [C_1], \ldots, [D_{i,j}] \bullet [C_u])$
$[\boldsymbol{\gamma}_i] \leftarrow \sum_{j=1}^{w} [\boldsymbol{\tau}_j]$
return $[\boldsymbol{\gamma}_i]$

Algorithm 3 Indicator($m, [k]$) $\rightarrow [\boldsymbol{k}]$

Require: m specifies the number of attributes, and k is a positive integer in $\{1, \ldots, m\}$.
for $i \leftarrow 1$ to m **do**
  $[\boldsymbol{k}_i] \leftarrow$ Equal($i, [k]$) ▷ check if k is equal to i
return $[\boldsymbol{k}]$

three branches. Due to space limitation, we only present part of it. According to the figure, Γ_1 represents A_2 as the chosen root of Γ, then Γ_{21}, Γ_{22} and Γ_{23} correspond to the sub-trees rooted at A_1, A_3 and A_5 respectively. Γ_1 is represented using A_2's indicator vector 01000. Let a tuple $t = [2, 3, 3, 1, 1]$ represented as a 3-by-5 matrix:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

t_i indicates the i-th row of the matrix, and $|t_i|$ gives the number of attributes. The number of rows in t is denoted by w specifying the domain size of each attribute. In this example, $|t_i| = 5$ and $w = 3$.

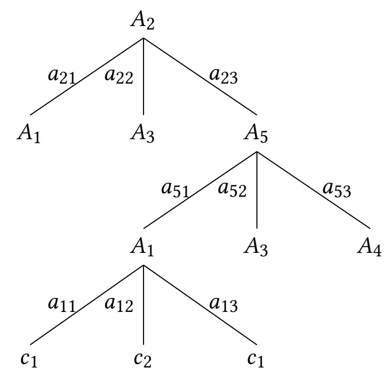


Figure 1: A partial decision tree

Algorithm 4 Predict($[t], [\Gamma]$) $\rightarrow [c]$

Require: $[t]$ secret shares of a tuple being classified, $[\Gamma]$ secret shares of the decision tree, and w denotes the number of attributes.
1: **if** $|[\Gamma]| = 1$ **then**
2:   **return** $[\Gamma]$
3: $[\Gamma_1], [\Gamma_2] \leftarrow [\Gamma]$
4: **for** $i \leftarrow 1$ to w **do**
5:   $[\boldsymbol{f}_i] \leftarrow [t_i] \bullet [\Gamma_1]$
6: **for** $i \leftarrow 1$ to w **do**
7:   $[\boldsymbol{c}_i] \leftarrow$ Predict($[t], [\Gamma_{2i}]$)
8: $[c] \leftarrow [\boldsymbol{f}] \bullet [\boldsymbol{c}]$
9: **return** $[c]$

The prediction protocol is given by Protocol 4. It derives the class label of a secretly shared record t recursively. The tree Γ is also secretly shared based on its smallest elements. For example, Γ_1 is secretly shared component-wise of A_2's indicator vector at the top level of the tree. This recursively applies to the other levels. At the leaf level, the class label c is secretly shared. The tree structure, a complete w-nary tree, is preserved, so that we can easily check if Γ is a tree or a leaf node. At step 1 of the protocol. $|[\Gamma]| = 1$ indicates that $[\Gamma]$ is a leaf node, and it is returned as the leaf label. Otherwise, $[\Gamma]$ is decomposed into two components $[\Gamma_1]$ and $[\Gamma_2]$. At steps 4-5, the protocol retrieves the correct attribute at the current root whose values serve as flags to indicate the right prediction path. At steps 6-7, the protocol makes w recursive calls on the sub-trees. At the end, the results from each sub-recursive calls are combine to produce the shares of the class label. The key observation is that there is only one path in the tree whose associated $\boldsymbol{f}_i$ values are all one. As a result, the results returned from all other paths are 0, and the result from the correct path is preserved.

4 EXPERIMENTAL RESULTS

We empirically evaluate the model accuracy and efficiency of our proposed ODP+FMPC-ID3 protocol (Algorithm 1) comparing to the existing solution CDP+PMPC-ID3 protocol (Algorithm 7 in Appendix B). We also implemented MPC-ID3 which offers identical accuracy as the non-secure ID3 and serves as the baseline for accuracy evaluation. All implementation were written in python, and we used the MPyC library [35] for basic MPC operations. In our experiments, we use the following three real-world datasets: Titanic [28], Heart [20] and Adult [3]. All datasets have binary class labels. Additional statistics about each dataset is given in Table 1 of Appendix C.4. For the utility evaluation, all experiments are performed over 5 different 80-20 train-test splits. We measure the accuracy and ROC AUC on the test set.

To evaluate protocol efficiency, we run secure 3-party computations where all experiments are repeated at least 3 times on a 80-20 train-test split. We measure the runtime and the network traffic for training and inference. The network traffic refers to the total amount of messages each party received during the computation. Additional implementation details are given in Appendix C.4.

4.1 Accuracy Evaluation

To evaluate accuracy, we mainly focus on three parameters: privacy loss ϵ, tree depth and DP mechanisms (Laplace vs. Exponential).

4.1.1 Varying Privacy Loss. We evaluate the accuracy and the ROC AUC of different variants of the ID3 algorithm. Figure 2 shows the relationship between model utility and the privacy loss ϵ with various datasets under the fixed tree depth. For Titanic, Heart, and Adult, the maximum depths are set to 4, 3, and 5, respectively. Regarding the privacy loss parameter, for Titanic and Heart, the values of ϵ are varied from 0.2 to 2.0. For Adult, a larger dataset, the values of ϵ are varied from 0.005 to 1.0. We did not use smaller ϵ values for Titanic and Heart because ϵ less than 0.2 would make the accuracy below 0.5 which causes the model useless.

The plots compare the accuracy of our proposed solution ODP-FMPC-ID3 against that of the existing solution CDP-PMPC-ID3 under the same ϵ-DP guarantee, benchmarking with MPC-ID3. Note that the accuracy of MPC-ID3 matches that of non-private centralized learning. As observed in Figure 2, the accuracy of our solution consistently performs better. For example, when $\epsilon = 0.2$, our solution produced an accuracy of 0.71 and ROC AUC of 0.71 for Heart. Whereas, under the same ϵ, the existing solution generated an accuracy of 0.575 and ROC AUC of 0.572. This amounts to 23.5% and 24.1% improvements respectively. For the Adult dataset, $\epsilon = 0.005$, our solution produced an accuracy of 0.755 and ROC AUC of 0.65. However, under the same ϵ, the existing solution only generated an accuracy of 0.585 and ROC AUC of 0.525. Thus, our solution amounts to 29.1% and 23.8% improvements. Additionally, for Adult dataset, when ϵ is between 0.1 and 1.0, our approach even outperforms the non-private centralized learning. This would be the effect of the randomness introduced by DP, acting as regularization. In general, our solution performs better as ϵ gets smaller.

4.1.2 Varying Tree Depth. We also evaluate the model accuracy with different tree depths, which is shown in Figure 3. We measure ROC AUC under a fixed ϵ while varying the tree depths from 2 to 5. We use $\epsilon = 0.2, 0.2, 0.1$ for the three datasets Titanic, Heart, and Adult respectively. The performance trends are the same for other ϵ values and the accuracy metric.

In DP decision trees, the tree depth affects model accuracy. As a tree becomes deeper, training samples at each leaf node become very small and thus can hinder accuracy. Because DP leaf updates are based on the frequency of the class counts, smaller counts are susceptible to DP noise. As seen in Figure 3, since Titanic and Heart are smaller datasets, increasing the tree depth greatly lowers the model accuracy. However, for Adult dataset, since the data size is much bigger, under the chosen $\epsilon = 0.1$, increasing the tree depth from 2 to 5 does not degrade the accuracy. It simply requires a deeper tree to learn a model. On the other hand, this is not the case when using much smaller values of ϵ.

As also shown in Figure 3, our solution consistently provides a better accuracy for each of the maximum tree depths. In the existing solution, the privacy loss budget is split across different heights of tree. Thus, each node of the tree gets a budget of $\epsilon/(h+1)$, where h is the maximum depth of the tree. Whereas, for our proposed approach, the full privacy budget ϵ is used for each leaf node. As a result, our solution is more robust to the maximum depth.

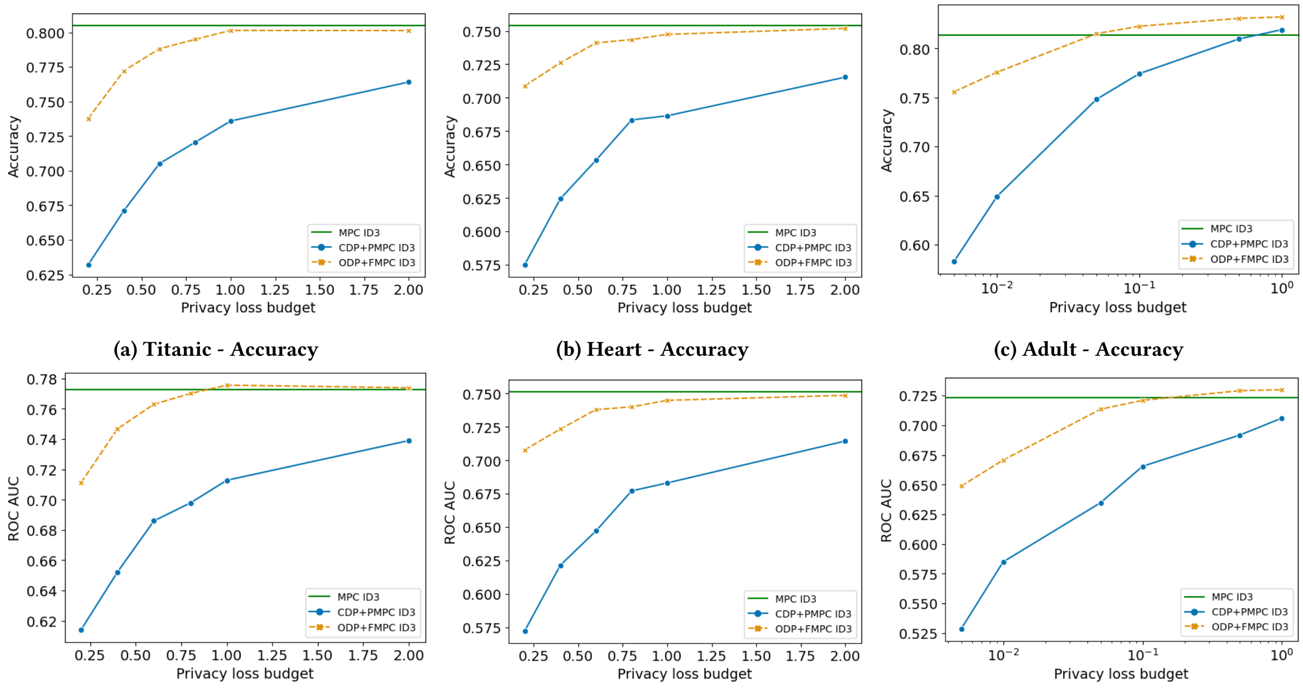
4.1.3 Comparison between Laplace and Exponential. In the previous results, we use the Laplace mechanism for DP leaf updates for both approaches. Here, we additionally show the model accuracy when using the Exponential mechanism. While Fletcher et al. suggested an advantage of the Exponential mechanism against the Laplace mechanism for multi-class random forest models [12], as far as we know, there is no consensus on which DP algorithm performs the best. In fact, recent works use the Laplace mechanism for leaf updates [7, 18].

Figure 4 compares the accuracy of the Laplace mechanism and that of the Exponential mechanism, varying privacy loss ϵ from 0.2 to 2.0 for Titanic and Heart and 0.005 to 1.0 for Adult. Similar to Figure 2, the maximum depth is fixed for each plot, where $h = 4, 3, 5$ for Titanic, Heart, and Adult respectively. For both approaches, the Laplace mechanism provides a better accuracy than the Exponential mechanism by up to 5%. While with large values of ϵ, there is no huge difference between the two DP mechanisms, with small values of ϵ, the Laplace mechanism consistently outperforms the Exponential mechanism.

4.2 Efficiency Evaluation

We show the runtime and the network traffic of training ID3 for our solution and the existing work under a fixed ϵ of 1.0 in Figure 5. We varied the maximum tree depth from 2 to 5 for each dataset. For each approach, we have the two variants of the leaf update method, the Laplace mechanism-based update and the Exponential mechanism-based update.

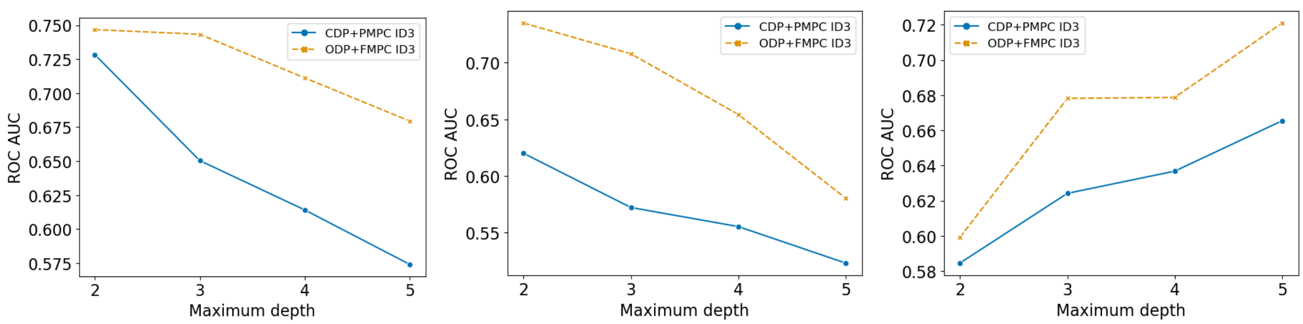
Overall, our solution is computationally more expensive for both runtime and network traffic. It relies on the full MPC-based ID3 functionality, which is computationally more expensive than the partial MPC-based ID3 used in the existing work. Especially, our approach builds a completely oblivious tree where every non-leaf node has w branches, where w is the maximum domain size. Thus, the runtime and the network traffic greatly increase as the depth of the tree increases. This is the tradeoff between accuracy



(a) Titanic - Accuracy (b) Heart - Accuracy (c) Adult - Accuracy

(d) Titanic - ROC AUC (e) Heart - ROC AUC (f) Adult - ROC AUC

Figure 2: Model accuracy vs. privacy loss budget ϵ



(a) Titanic (b) Heart (c) Adult

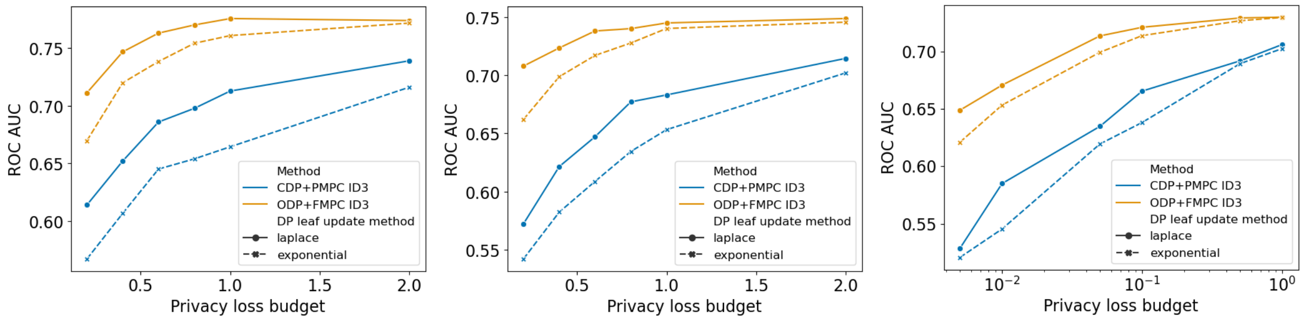
Figure 3: Model accuracy vs. tree depth

and efficiency, where a high accuracy observed in the enhanced framework was achieved by the computationally expensive full MPC protocol that can hide the model. Building efficient MPC-based decision trees are independent of our work, and it is still an active research area.

In addition, Figure 5 compares the efficiency of the MPC-based Laplace mechanism and the MPC-based Exponential mechanism (both protocols are presented in Appendix C.3). While there was no clear distinction for the existing approach, we observed that for our solution, the Laplace mechanism incurs less runtime and network traffic than the Exponential mechanism. The gap becomes very noticeable as we increase the maximum depth. The MPC-based Exponential mechanism requires a number of MPC operations, including a very costly secure exponentiation evaluation. The detailed cost analysis on MPC can be found in [10].

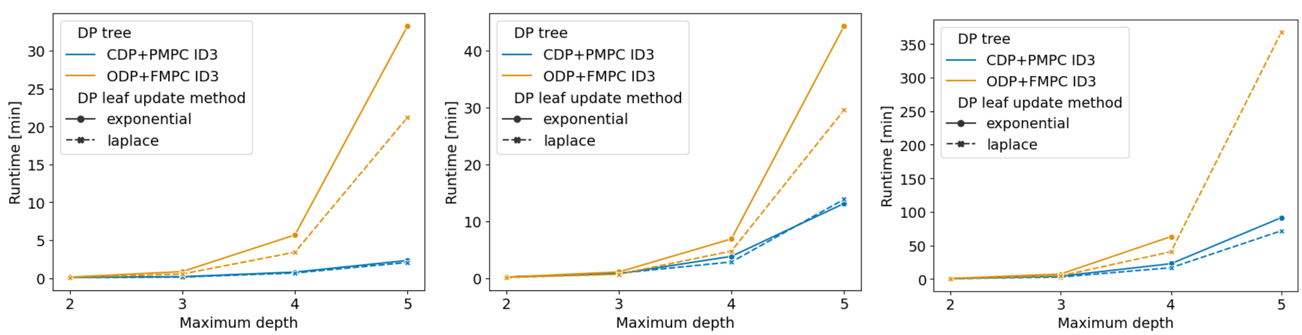
5 CONCLUSION AND FUTURE WORK

In this paper, we proposed a new approach for building a differentially private decision tree. Utilizing MPC to its max, the decision tree can be kept hidden even during model prediction. As a result, our solution achieves (1) data privacy, (2) high model accuracy, (3) model confidentiality and (4) query privacy. Furthermore, our solution can be naturally extended to random forest model, and this extension is discussed in Appendix D.

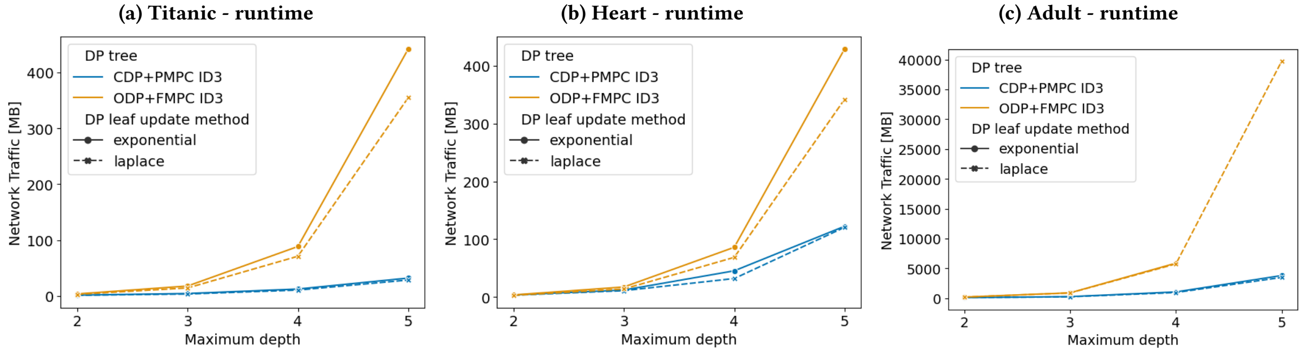


(a) Titanic (b) Heart (c) Adult

Figure 4: Accuracy comparison on different DP leaf update methods



(a) Titanic - runtime (b) Heart - runtime (c) Adult - runtime



(d) Titanic - network traffic (e) Heart - network traffic (f) Adult - network traffic

Figure 5: Efficiency vs. tree depth

On the other hand, the main drawback of our solution is its efficiency. As a future research direction, we will explore more efficient ways to construct a fully secure protocol for generating decision trees. One potential solution is to design customized MPC functionalities for our specific ML application. For example, since secure comparison has been used extensively in the sub-protocols, a more efficient secure comparison can greatly improve the overall protocol efficiency. In addition, the efficiency of the prediction protocol could be improved by directly using secret shares to represent attributes for the internal nodes instead of the vector indicator representation. However, the challenge is to develop an efficient and customized equality test or attribute selection sub-protocol. We will also investigate a tighter bound on DP guarantee of our random forest solution. In our proof of Claim D.1, we assumed the worst case. By utilizing the additional randomness among the sub-samples, we may be able to derive a more precise DP bound for the random forest.

REFERENCES

[1] Mark Abspoel, Daniel Escudero, and Nikolaj Volgushev. 2020. Secure training of decision trees with continuous attributes. *Cryptology ePrint Archive* (2020).

[2] Borja Balle, Gilles Barthe, and Marco Gaboardi. 2018. Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/

3b5020bb891119b9f5130f1fea9bd773-Paper.pdf

[3] Barry Becker and Ronny Kohavi. 1996. Adult. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5XW20.

[4] Mariusz Bojarski, Anna Choromanska, Krzysztof Choromanski, and Yann LeCun. 2014. Differentially-and non-differentially-private random decision trees. *arXiv preprint arXiv:1410.6973* (2014).

[5] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (Leveled) Fully Homomorphic Encryption without Bootstrapping. *ACM Trans. Comput. Theory* 6, 3, Article 13 (jul 2014), 36 pages. https://doi.org/10.1145/2633600

[6] Leo Breiman and JH Friedman. 1984. RA Olshen und CJ Stone. *Classification and regression trees* (1984).

[7] Shorya Consul and SA William. 2021. Differentially private random forests for regression and classification. *Association for the Advancement of Artificial Intelligence* (2021).

[8] Sebastiaan De Hoogh, Berry Schoenmakers, Ping Chen, and Harm op den Akker. 2014. Practical secure decision tree learning in a teletreatment application. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18*. Springer, 179–194.

[9] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[10] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. 2014. Differentially private data aggregation with optimal utility. In *Proceedings of the 30th Annual Computer Security Applications Conference*. 316–325.

[11] Sam Fletcher and Md Zahidul Islam. 2015. A differentially private random decision forest using reliable signal-to-noise ratios. In *AI 2015: Advances in Artificial Intelligence: 28th Australasian Joint Conference, Canberra, ACT, Australia, November 30–December 4, 2015, Proceedings 28*. Springer, 192–203.

[12] Sam Fletcher and Md Zahidul Islam. 2017. Differentially private random decision forests using smooth sensitivity. *Expert systems with applications* 78 (2017), 16–31.

[13] Arik Friedman and Assaf Schuster. 2010. Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 493–502.

[14] Craig Gentry. 2009. Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing* (Bethesda, MD, USA) *(STOC '09)*. Association for Computing Machinery, New York, NY, USA, 169–178. https://doi.org/10.1145/1536414.1536440

[15] Oded Goldreich. 2004. *The Foundations of Cryptography*. Vol. 2. Cambridge University Press, Chapter General Cryptographic Protocols. http://www.wisdom.weizmann.ac.il/~oded/PSBookFrag/prot.ps

[16] Koki Hamada, Dai Ikarashi, Ryo Kikuchi, and Koji Chida. 2021. Efficient decision tree training with new data structure for secure multi-party computation. *arXiv preprint arXiv:2112.12906* (2021).

[17] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. 2019. Diffprivlib: the IBM differential privacy library. *ArXiv e-prints* 1907.02444 [cs.CR] (July 2019).

[18] Jun Hou, Qianmu Li, Shunmei Meng, Zhen Ni, Yini Chen, and Yaozong Liu. 2019. DPRF: a differential privacy protection random forest. *Ieee Access* 7 (2019), 130707–130720.

[19] Geetha Jagannathan, Krishnan Pillaipakkamnatt, and Rebecca N Wright. 2009. A practical differentially private random decision tree classifier. In *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 114–121.

[20] Steinbrunn William Pfisterer Matthias Janosi, Andras and Robert Detrano. 1988. Heart Disease. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C52P4X.

[21] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends® in Machine Learning* 14, 1–2 (2021), 1–210. https://doi.org/10.1561/2200000083

[22] Nhan Khanh Le, Yang Liu, Quang Minh Nguyen, Qingchen Liu, Fangzhou Liu, Quanwei Cai, and Sandra Hirche. 2021. Fedxgboost: Privacy-preserving xgboost for federated learning. *arXiv preprint arXiv:2106.10662* (2021).

[23] Qinbin Li, Zhaomin Wu, Zeyi Wen, and Bingsheng He. 2020. Privacy-Preserving Gradient Boosting Decision Trees. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 01 (Apr. 2020), 784–791. https://doi.org/10.1609/aaai.v34i01.5422

[24] Samuel Maddock, Graham Cormode, Tianhao Wang, Carsten Maple, and Somesh Jha. 2022. Federated Boosted Decision Trees with Differential Privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2249–2263.

[25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. https://proceedings.mlr.press/v54/mcmahan17a.html

[26] Frank McSherry and Kunal Talwar. 2007. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 94–103.

[27] Shagufta Mehnaz, Sayanton V. Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. 2022. Are Your Sensitive Attributes Private? Novel Model Inversion Attribute Inference Attacks on Classification Models. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 4579–4596. https://www.usenix.org/conference/usenixsecurity22/presentation/mehnaz

[28] Vinicius Barbosa Paiva. 2020. The Complete Titanic Dataset. https://www.kaggle.com/datasets/vinicius150987/titanic3.

[29] European Parliament and of the Council. 2016. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). https://eur-lex.europa.eu/eli/reg/2016/679/oj

[30] Abhijit Patil and Sanjay Singh. 2014. Differential private random forest. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2623–2630.

[31] Sikha Pentyala, Davis Railsback, Ricardo Maia, Rafael Dowsley, David Melanson, Anderson Nascimento, and Martine De Cock. 2022. Training differentially private models with secure multiparty computation. *arXiv preprint arXiv:2202.02625* (2022).

[32] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. 2015. Phasing: Private Set Intersection Using Permutation-based Hashing. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 515–530. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pinkas

[33] J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning* 1 (1986), 81–106.

[34] J Ross Quinlan. 1993. Program for machine learning. *C4. 5* (1993).

[35] Berry Schoenmakers. 2018. MPyC: Multiparty Computation in Python. https://www.win.tue.nl/ berry/mpyc/.

[36] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership Inference Attacks Against Machine Learning Models. In *2017 IEEE Symposium on Security and Privacy (SP)*. 3–18. https://doi.org/10.1109/SP.2017.41

[37] K.K.A. Thissen. 2019. *Achieving Differential Privacy in Secure Multiparty Computation*. Ph. D. Dissertation. Master's Thesis, Technische Universiteit Eindhoven, Eindhoven.

[38] Zhihua Tian, Rui Zhang, Xiaoyang Hou, Jian Liu, and Kui Ren. 2020. Federboost: Private federated learning for gbdt. *arXiv preprint arXiv:2011.02796* (2020).

[39] Rui Wang, Oğuzhan Ersoy, Hangyu Zhu, Yaochu Jin, and Kaitai Liang. 2022. Feverless: Fast and secure vertical federated learning based on xgboost for decentralized labels. *IEEE Transactions on Big Data* (2022).

[40] Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. 2020. Privacy Preserving Vertical Federated Learning for Tree-Based Models. *Proc. VLDB Endow.* 13, 12 (jul 2020), 2090–2103. https://doi.org/10.14778/3407790.3407811

[41] Bangzhou Xin, Wei Yang, Shaowei Wang, and Liusheng Huang. 2019. Differentially private greedy decision forest. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2672–2676.

[42] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *ACM Trans. Intell. Syst. Technol.* 10, 2, Article 12 (jan 2019), 19 pages. https://doi.org/10.1145/3298981

[43] Andrew C. Yao. 1986. How to Generate and Exchange Secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. 162–167. https://doi.org/10.1109/SFCS.1986.25

A BACKGROUND

We consider a private dataset D where each column and row correspond to an attribute and a sample respectively. There are m non-class attributes $\mathcal{A} = \{A_0, A_1, \cdots, A_m\}$ and the class attribute is denoted by A_0. The domain of each attribute A_i is finite, denoted as $dom(A_i)$. We assume that all attributes are composed of discrete variables. Namely, continuous attributes are discretized. The discretization is a common pre-processing technique used in machine learning and data science fields. In addition, the schema (i.e., number of attributes, attribute domains, domain sizes) is considered public. For $v \in dom(A_i)$, we denote $D_{A_i=v}$ as a subset of samples whose attribute A_i is equal to v. In addition, we denote $|D_{A_i=v}|$ as the number of samples in $D_{A_i=v}$. We may write $N_v = |D_{A_i=v}|$ if the context is clear.

A.1 Differential Privacy

We consider a dataset D that is a collection of individual data coming from the universe of all possible datasets $\mathcal{D}$. Differential Privacy (DP) [9] has become the de facto standard for privacy protection. DP is a mathematical privacy definition that guarantees the output of the mechanism will not differ significantly between any two neighboring datasets. We say D and D' are neighbors, denoted as $D \sim D'$, if they differ at most by one individual. It is formally defined as follows.

Definition A.1 ((ϵ, δ)-Differential Privacy). *A randomized mechanism $\mathcal{M}$ is (ϵ, δ)-differentially private if for all output $O \in Range(\mathcal{M})$ and for any two neighboring datasets $D \sim D' \in \mathcal{D}$,*

$$Pr[\mathcal{M}(D) = O] \leq e^{\epsilon} Pr[\mathcal{M}(D') = O] + \delta.$$

When $\delta = 0$, we say $\mathcal{M}$ satisfies ϵ-DP.

The privacy parameter ϵ is referred to as the privacy loss budget where a smaller value of ϵ gives more privacy protection but can lower data utility. DP is typically achieved by a noise-addition mechanism where a carefully calibrated noise is added to the outcome of a function f. The magnitude of the noise is controlled by the privacy parameter ϵ and function sensitivity Δ_f defined below.

Definition A.2 (Sensitivity). *The sensitivity Δ_f of a function $f : \mathcal{D} \to \mathbb{R}$ is defined as $\Delta_f = \max_{\forall D \sim D'} |f(D) - f(D')|$.*

A common mechanism for outputting a real number is to use the Laplace mechanism where we add a noise, denoted as $Lap(b)$, sampled from the Laplacian distribution with the mean of 0 and a scale of $b = \Delta_f/\epsilon$. We denote $Lap(\cdot)^l$ as l independent Laplacian noise. It is well known that the following Laplace mechanism satisfies ϵ-DP.

Definition A.3 (Laplace mechanism [9]). *Given $f : \mathcal{D} \to \mathbb{R}^l$, the mechanism that outputs $f(D) + Lap(\Delta_f/\epsilon)^l$ satisfies ϵ-DP.*

For a private selection over discrete values or items, the Exponential mechanism can be used. Given a finite set of candidates $\mathcal{R}$ and an associated quality score function $q : (\mathcal{D} \times \mathcal{R}) \to \mathbb{R}$, the Exponential mechanism aims to return a candidate $r \in \mathcal{R}$ that approximately maximize $q(D, r)$ while enforcing DP. The quality function is typically a way of measuring the quality of the candidate r with respect to data D, where higher scores are better. It induces a probability distribution over the output domain that favors high scoring outcomes. In other words, candidates that have higher scores are exponentially more likely to be chosen. It is well known that the following Exponential mechanism satisfies ϵ-DP.

Definition A.4 (Exponential mechanism [26]). *Given a quality (or scoring) function $q : (\mathcal{D} \times \mathcal{R}) \to \mathbb{R}$, the mechanism $\mathcal{M}$ defined as follows satisfies ϵ-DP.*

$$Pr[\mathcal{M}(D) = a] \propto exp\left(\frac{\epsilon q(D, a)}{2\Delta_q}\right).$$

DP mechanisms have the following properties: the post-processing property states that all subsequent analyses on outputs produced by ϵ-DP mechanism satisfy ϵ-DP, without degrading privacy loss. Additionally, the sequential composition property implies that if we answer two query sets under ϵ_1 and ϵ_2, respectively, the combined answers satisfy $(\epsilon_1 + \epsilon_2)$-DP. The parallel composition says that given a dataset D, a disjoint partition $D = D_1 \cup, \ldots, \cup D_k$ and ϵ-DP mechanism $\mathcal{M}$, the mechanism $\mathcal{M}'(D) = (\mathcal{M}(D_1), \ldots, \mathcal{M}(D_k))$ satisfies ϵ-DP.

A.2 Tree-Based Model

We consider an ID3-based classification algorithm[33]. However, our ideas are applicable to other variants such as CART [6], C4.5 [34] or random forests. The general framework of these algorithms are the same where they are all greedy algorithms, finding the optimal decision and recursively building a tree.

A.2.1 The ID3 Algorithm. The key steps of ID3 are given in Algorithm 5, where a tree is built from root to leaves recursively. It first checks if some stopping criterion is satisfied (e.g., attribute set is empty, tree reaches the maximum depth, the number of samples is less than a threshold). If any stopping criterion is satisfied, the algorithm returns a leaf node with the class of majority samples. Otherwise, it determines the best splitting attribute $A_* \in \mathcal{A}$ for the parent node, according to some quality measure q, and splits the samples with that attribute. The same process is applied recursively on each subset of the samples.

Algorithm 5 Tree($D, \mathcal{A}, C$) $\to T$

```
Require: Dataset D, set of attributes 𝒜, class attribute C
Create a node T
if stopping criterion is met then
    return T with the majority class label argmax_c |D_{C=c}|
Find the best attribute A_* = argmax_{A∈𝒜} q(D, A) according to
quality function q
Associate A_* with T
for a_j ∈ dom(A_*) do
    T_j ← Tree(D_{A_*=a_j}, 𝒜 − A_*, C)
    Assign T_j as the j-th child of T
return T
```

A.2.2 The DP-ID3 Algorithm. A DP version of such algorithm aims to limit the individual-level privacy leakage about the training dataset that the decision tree learns. In a typical DP decision tree learning problem, the tree is assumed to be released to public. Thus, to provide a DP guarantee, a noise is added whenever a sensitive data D is queried to construct the tree. While there are many strategies, key components that require adding some form of noise or DP consideration are: node splitting (i.e., which attribute to split a node with), and leaf updates (i.e., compute leaf weights or labels for prediction).

A.2.3 DP Node Splitting. To choose the best split attribute with DP guarantee, the most common and efficient solution is to use the Exponential mechanism where the best attribute is selected with a high probability based on a specified quality metric [13]. For example, the attribute that has the lowest Gini index is chosen with a high probability. More formally, given a dataset D, a set of attributes $\mathcal{A}$, a quality function $q : \mathcal{D} \times \mathcal{A} \to \mathbb{R}$, the best attribute $A_{best} \in \mathcal{A}$ is outputted with probability $\propto \exp\left(\frac{\epsilon q(D, A_{best})}{2\Delta_q}\right)$.

Algorithm 6 DPTree$(D, \mathcal{A}, C, h, \epsilon) \rightarrow T$

Require: Private dataset D, set of attributes $\mathcal{A}$, class attribute C, tree depth h, privacy budget per query ϵ

```
Create a node T
if A = ∅ or h = 0 then
    Decide majority class c ∈ dom(C) via DP Leaf Updates
    return T with the majority class label c
Determine the best attribute A* ∈ A via DP Node Splitting
Associate A* with T
for a_j ∈ dom(A*) do
    T_j ← DPTree(D_{A*=a_j}, A − A*, C, h − 1, ε)
    Assign T_j as the j-th child of T
return T
```

A.2.4 DP Leaf Updates. We consider two common DP approaches of deciding the majority label of each leaf node. The first method is based on private counting using the Laplace mechanism [4, 7, 11, 18, 19, 30] and the second method is based on private selection using the Exponential mechanism [12, 17, 41], detailed below.

Private Counting. The first approach is to use the idea of private counting where for each leaf node, we generate noisy class counts using the Laplace mechanism and determine the majority class label based on the resulting noisy counts. Namely, the majority label is selected by computing $\text{argmax}_{c \in dom(C)}\{|D_{C=c}| + Lap(1/\epsilon)\}$.

Private Selection. The second approach is to use the Exponential mechanism to determine the majority label using counts as a quality function. Given a dataset D, a class attribute C, a counting query $q : \mathcal{D} \times dom(C) \rightarrow \mathbb{N}$, the majority class label $c_{max} \in dom(C)$ is selected with probability $\propto \exp\left(\frac{\epsilon q(D, c_{max})}{2\Delta_q}\right)$, where $\Delta_q = 1$ (the sensitivity of a counting query is 1).

In Algorithm 6, we consider a data independent stopping criterion, which checks if the attribute set is empty or the tree reaches the predefined depth, to avoid spending additional privacy budget.

B BASIC MPC+DP FRAMEWORK

Given a secret-shared data, a natural way of constructing a tree with DP guarantees is to train a DP tree-based model as in Algorithm 6 using an MPC framework. We provide a general framework of building a DP decision tree with MPC in Algorithm 7. A trained model including split attributes and output class labels are safely revealed in a plaintext format with DP guarantees during the protocol execution. Since the resulting model satisfies DP, it can be shared with a third party in plaintext without degrading privacy, but this does not achieve model confidentiality. Although the idea presented in Algorithm 7 is not new, it is still challenging to implement the procedure properly using MPC. In addition, without implementing algorithm, we will not be able to compare its model accuracy and run-time efficiency with our solution. Next we present our implementation details.

B.1 Centralized DP Decision Tree Learning with Partial MPC

First, we provide an overview of Algorithm 7, which trains a DP decision tree model on a secret shared data $[D]$ with an MPC framework. At each non-leaf node, quality metrics (i.e., the max operator) of candidate attributes $\mathcal{A}$ are computed using MPC-Score. The best split attribute is decided under DP with MPC and revealed as a plaintext (Algorithm 8). With the best attribute A_*, the MPC-Partition protocol secretly partitions $[D]$ into $|dom(A_*)|$ disjoint datasets such that each data subset includes all samples whose $A_* = a$ for every $a \in dom(A_*)$. Note that $[D_{A_*=a}]$ is a symbolic notation: it does not mean the data are actually decomposed. It typically requires some form of secret indicators of whether a sample belongs to a certain node. Meanwhile, for each leaf node, frequency counts of each class label are computed using MPC-Count. With the resulting secretly shared counts, the majority class label is determined under DP with MPC and revealed as a plaintext (Algorithm 9).

A key observation is that any intermediate secretly shared values in which DP noise is incorporated are safely revealed in a plaintext format under a DP guarantee during a tree construction. We take this as an advantage to make the protocol as efficient as possible, which we will detail in the next section. Since we train a DP decision tree on top of MPC, the model accuracy provided by Algorithm 7 is theoretically the same as that of (non-MPC) DP decision tree provided by Algorithm 6. However, in practice, there might be some small discrepancy in accuracy due to the quantization of DP mechanism when implemented with a secret sharing scheme. MPC-Count, MPC-Partition and MPC-Score are left as black box operations. The actual instantiations will be described later.

Algorithm 7 CDP+PMPC-ID3$([D_{A_*=a_j}], \mathcal{A}, C, h, \epsilon)$

Require: Secretly shared dataset $[D]$, set of attributes $\mathcal{A}$, class attribute C, tree depth h, privacy budget per query ϵ

```
Create a node T
if A ≠ 0 or h = 0 then
    {[N_c]}_{c∈dom(C)} ← MPC-Count([D], C)
    c ← MPC-DPLeafUpdate({[N_c]}_{c∈dom(C)}, C, ε)
    return T with the majority class label c
for A_i ∈ A do
    [q_i] = MPC-Score([D], A_i)
A* ← MPC-DPNodeSplit({[q_i]}, ε, Δ = 1)
{[D_{A*=a_j}]}_{a_j∈dom(A*)} ← MPC-Partition([D], A*)
for a_j ∈ dom(A*) do
    T_j ← CDP+PMPC-ID3([D_{A*=a_j}], A − A*, C, h − 1, ε)
    Assign T_j as the j-th child of T
return T
```

B.2 DP Node Split and Leaf Update with MPC

Here we describe the details of DP node split and DP leaf update operations using an MPC protocol to hide the training data. We will observe the effect of DP on MPC protocols from the perspective of efficiency, and there is a key distinction between the Laplace mechanism and the Exponential mechanism in the efficiency aspect when incorporated into an MPC tree building process, especially for the DP leaf update.

B.2.1 *DP Mechanisms with MPC.*

The implementation of DP mechanisms, including the Laplace mechanism and the Exponential mechanism with secret-sharing based MPC, has been studied in prior works [10, 37], which we will use for our implementation. We denote MPC-LM($[x], b, \epsilon, \Delta$) as an MPC version of Laplace mechanism where secretly shared value of Laplacian noise $Lap(b)$ is sampled and then added to a secretly shared value of x to generate a secretly shared value of $x + Lap(b)$. No one learns any information about either input x or noise $Lap(b)$.

In addition, we denote MPC-EM($\{[q_1], \dots, [q_d]\}, \epsilon, \Delta$) as an MPC-version of the Exponential mechanism, which privately selects the i-th item among d items based on their associated secret share values of the quality scores $\{q_1, \dots, q_d\}$ of a function q. According to Definition A.4, the i-th item is returned with the probability proportional to $\exp(\epsilon q_i / 2\Delta)$. The output index i is secretly shared. Detailed implementations of both DP-mechanisms are given in Appendix C.3.

B.2.2 *DP Node Split.*

Algorithm 8 performs a DP node split using an MPC framework for a non-leaf node. Given secret shares of quality metrics, computed based Equation 1, the Exponential mechanism MPC-EM secretly and noisily chooses the approximate best attribute. Recall that the sensitivity of the max operator is 1, i.e., $\Delta = 1$. By executing the MPC-Recover, the resulting attribute is safely revealed in the protocol as plaintext since it satisfies DP without further degrading privacy.

B.2.3 *DP Leaf Update: Exponential Mechanism vs Laplace Mechanism.*

Algorithm 9 shows DP leaf update with MPC, given secret shares $\{[N_c]\}_{c \in dom(C)}$ of class frequency counts at a leaf node. The algorithm includes two approaches, one is based on the Laplace mechanism and the other is based on the Exponential mechanism. A key observation here is that for the Laplace mechanism based approach, after computing secret-shared DP counts, we can reveal the DP counts safely. As a result, we can compute the subsequent argmax operation in the clear. We emphasize that revealing the DP counts will not degrade the privacy loss. Furthermore, introducing DP into MPC allows us to eliminate an expensive MPC argmax operation that would be required in an MPC-only computation. Note that this does not mean the DP-MPC approach is more computationally efficient than the MPC-only approach as it has an extra complexity of computing the Laplace noise. However, since the Laplace noise generation is data-independent, when considering an online-offline setting, the DP-MPC approach would provide a better online efficiency than the MPC-only approach. On the other hand, the Exponential mechanism-based leaf update does not provide such an efficiency gain as it is data-dependent. Since the Laplace mechanism and the Exponential mechanism for leaf updates incur the same privacy cost, one may prefer the Laplace mechanism for efficiency. We will evaluate this in our experiments.

Algorithm 8 MPC-DPNodeSplit($\mathcal{A}, \{[q_i]\}, \epsilon, \Delta$)

1: $[i_*]$ = MPC-EM($\{[q_i]\}, \epsilon, \Delta$)
2: i_* = MPC-Recover($[i_*]$)
3: **return** A_{i_*}

Algorithm 9 MPC-DPLeafUpdate$_{LM}$($\{[N_c]\}_{c \in dom(C)}, C, \epsilon$)

Require: Set of class frequency counts$\{[N_c]\}_{c \in dom(C)}$, class attribute C, privacy parameter ϵ
1: **for** $c \in dom(C)$ **do**
2: $[\tilde{N_c}]$ = MPC-LM($[N_c], 1/\epsilon$)
3: $\tilde{N_c}$ = MPC-Recover($[\tilde{N_c}]$)
4: $c_* = \arg\max_{c \in dom(C)}(\tilde{N_c})$
5: **return** c_*

Algorithm 10 MPC-DPLeafUpdate$_{EM}$($\{[N_c]\}_{c \in dom(C)}, C, \epsilon$)

Require: Set of class frequency counts$\{[N_c]\}_{c \in dom(C)}$, class attribute C, privacy parameter ϵ
1: $[i_*]$ = MPC-EM($\{[N_c]\}_{c \in dom(C)}, \epsilon, 1$)
2: i_* = MPC-Recover($[i_*]$)
3: **return** c_{i_*}

B.3 Instantiation

We adopt some key ideas from the MPC ID3 construction of Protocol 4.1 by Hoogh et al. [8] to instantiate Algorithm 7. As mentioned before, The protocol completely hides the training data except for the information leaked from a tree output. Since the protocol reveals a tree (i.e., attribute splits and output class labels), it is the perfect MPC protocol to be applied to Algorithm 7 where a constructed DP tree can be safely disclosed. In general, Algorithm 7 can be constructed with any MPC decision tree protocols in combination with the MPC-DP mechanisms to produce a tree as plaintext.

A secret shared dataset $[D]$ is represented as secret shares of one hot encodings of length n for attribute A. That is, D_A is one-hot encoded into n-by-$|dom(A)|$ binary matrix. The instantiation of MPC-Count, MPC-Score and MPC-Partition simply follow the construction of Protocol 4.1 in [8]. Note that we use the max operator defined in Equation 1 for MPC-Score. Thus, the operation, which does not require a division, is much simpler and efficient than computing Gini index used in Protocol 4.1.

C ADDITIONAL DETAILS OF OUR PROPOSED SECURE ID3 PROTOCOL

C.1 Data Pre-processing

Before executing the model generation protocol, each data owner needs to locally pre-process its data. We assume that data owners apply consistent pre-processing strategies, such at discretization, handling missing values, etc. In addition, data integration at the servers can be tricky. Under the vertical partition scheme, the data owners generate one hot-encodings for their attributes and then secretly share the encodings with the three computing servers. The servers need to store the shares in the same ordering which can be guaranteed easily even if they do not know the data schema and attribute domains.

Under horizontal partition, it becomes more challenging to integrate the shares depending whether or not the schema is known. If we assume the schema (including each attribute domain) is known to the participating parties, then each party produces the one-hot encoding based on its own data. When locally producing the encodings for each attribute, the parties follow certain ordering of the attribute values. For example, the attribute may be sorted alphabetically and then map them to successive non-negative integers starting from 0 or 1. Suppose there are w distinct values for attribute A_i. Under horizontal partition, A_i is a feature owned by all parties. The key steps to generate and integrate the encodings of A_i are presented in Algorithm 11.

Algorithm 11 Encoding_Integration($\langle P_x, D_x, A, w\rangle, \langle S_z, \perp\rangle$) $\rightarrow$ $\langle S_z, [v]^z\rangle$

Require: P_x denotes one of the t data owner; that is, $x \in \{1, \dots, t\}$. D_x is the private dataset of P_x. A indicates an attribute index. w is the domain size of A.

```
{The following steps are performed by P_x}
for i = 1 to w do
    for t_j ∈ D_x do
        v_{i,j} = 0
        if t_j(A) = i then
            v_{i,j} = 1
[v]_x^1, [v]_x^2, [v]_x^3 = GenShares(P_x, v)
Send [v]_x^z to server S_z for z ∈ {1, 2, 3}
{The following steps are performed by S_z}
receive [v]_x^z from P_x for x ∈ {1, ..., t}
for i = 1 to w do
    [v_i]^z = [v_i]_1^z || ··· || [v_i]_t^z
return [v]^z
```

If the parties do not know the complete attribute domain, we need more advanced MPC protocols to align the one-hot encodings without leaking the additional domain knowledge. Developing efficient one-hot encoding alignment protocols are orthogonal to our work, and the basic idea is to utilize private set intersection protocols [32] to obliviously link the encodings that represent the same attribute value together. Moreover, our protocol hide the tree structure by enforcing each attribute has the same number of distinct values. Let w be the largest domain size among all attributes. Then each party needs to generate w one-hot encoding per attribute. For dummy attribute values, each party can simply generate zero vectors to represent the dummy values. When sorting the encodings, these zero vectors (per attribute) can be placed at the end of the encoding list for that specific attribute.

C.2 Privacy/Security Analysis

As stated before, our solution achieves (1) data privacy, (2) high model accuracy, (3) model confidentiality and (4) query privacy. For high model accuracy, we will demonstrate it in two ways: theoretically prove less noise is needed to achieve the same level of DP, and empirically justify the accuracy improvement using three real world datasets. For the others, we need to prove the following:

- The training data are not disclosed when building the tree.
- The final tree model and user queries are never disclosed during prediction.
- The tree is ϵ-DP.

The first two are straightforward to prove since we adopt the standard MPC functionalities to implement the MPC-DP mechanisms and the MPC-decision tree generation. Intuitively speaking, nothing is disclosed during model generation and prediction, except that a user obtains the final prediction result. As long as the servers follow the protocol and the secret sharing scheme is secure, so does our protocol. On the other and, since our way of combining DP and MPC is new, we need to prove the resulting protocol satisfies ϵ-DP. The key idea is to show that when the tree structure is hidden, adding less amount of noise to the leaf nodes is sufficient to achieve the same level of DP comparing to the existing solution where DP noises are also added to the attribute selection process for the internal nodes. Let T_1 and T_2 are two decision trees with the following properties:

- T_1: The tree structures are completely hidden including the leaf labels. In addition, before leaf labels are derived, noises are securely added to achieve ϵ-DP. During prediction, the users only learn the predicted result. The entire process can be achieved using MPC. (Note that T_1 may be more private than ϵ-DP since the leave labels are hidden.)
- T_2: The tree structures are known but noises are added during tree generation process to achieve ϵ-DP. More specifically, MPC techniques are adopted to compute the scores for each attribute. Using MPC, noises are securely added to the scores before selecting the best split or attribute. After the selection, the chosen attribute is disclosed. Then the process repeats to build the rest of the tree. For prediction purpose, the DP-tree can be shared with the users.

It is well-known that when an adversary has black-box access to a decision tree model, the adversary can reconstruct the tree with very high accuracy by using a number of queries and their prediction results. As a result, we may ask that even though T_1 is completely hidden from anyone, is it still possible for an adversary to reconstruct another tree $\hat{T}_1$ (with black-box access to T_1) that is less private than T_1 or T_2? The short answer is no because when deriving T_1, we followed the sequential composition theorem and all intermediate computations and results are implemented using MPC protocols. If $\hat{T}_1$ is less private than T_1 or T_2, then either the composition theorem is incorrect or the MPC protocol leaks non-negligible information. A more formal analysis is given next.

C.2.1 T_1 *is* ϵ*-Differentially Private.*

Here we prove that adding ϵ-DP noise at each leaf node independently guarantees T_1 is ϵ-DP. Let f be the mechanism that securely selects the best attribute for the internal nodes of T_1:

- $f([\vec{D}], [\vec{\Lambda}]) \rightarrow [\vec{D}_l], [\vec{D}_r], [\vec{\Lambda}_l], [\vec{\Lambda}_r], [a], [t_a]$

Without loss of generality, we assume each attribute splits the data into two partitions: left and right denoted by $\vec{D}_l$ and $\vec{D}_r$ respectively. All $\vec{D}$, $\vec{D}_l$ and $\vec{D}_r$ are binary vectors with the same size bounded by the size of the dataset, and the [ ] notation indicates the vectors are secretly shared component-wise. $\vec{\Lambda}$, $\vec{\Lambda}_l$ and $\vec{\Lambda}_r$ are also binary vectors with the same size determined by the number of attributes. The chosen attribute is denoted by a with threshold t_a. All inputs and outputs of f are secretly shared, and the height of the tree is fixed to a public parameter h indicating the tree has $0, 1, \ldots, h$ layers. $\vec{D}_l$ and $\vec{\Lambda}_l$ (or $\vec{D}_r$ and $\vec{\Lambda}_r$) are inputs to build the left (or right) branch of the sub-tree rooted at a. $[a]$ and $[t_a]$ are the actual output of each tree layer. Let g be a function for the leaf node:

- $g([\vec{D}]) \rightarrow [c]$

where $\vec{D}$ represents a set of tuples at the current leaf node and c is the class label for this node. The class label is chosen using the Laplace mechanism with privacy budget ϵ and sensitivity 1. Both $\vec{D}$ and c are secretly shared. Based on f and g, we define a layer function L as follows:

- $L^k = \langle f_1^k, \ldots, f_{2^k}^k \rangle$, for $k \in \{0, h-1\}$.
- $L^h = \langle g_1, \ldots, g_{2^h} \rangle$.

Each layer is a parallel composition of either the f or the g functions. Next we show f is 0-DP. Suppose $\vec{D}$ and $\vec{D}'$ represent two neighboring datasets with the same vector size. (Note that we can always make them the same size by adding a dummy entry to the smaller vector.) The f function produces the following outputs on the two neighboring datasets $\vec{D}$ and $\vec{D}'$:

- $f([\vec{D}], [\vec{\Lambda}]) \rightarrow [\vec{D}_l], [\vec{D}_r], [\vec{\Lambda}_l], [\vec{\Lambda}_r], [a], [t_a]$
- $f([\vec{D}'], [\vec{\Lambda}']) \rightarrow [\vec{D}'_l], [\vec{D}'_r], [\vec{\Lambda}'_l], [\vec{\Lambda}'_r], [a'], [t_{a'}]$

The output of f are secret shares from the same domain $\mathcal{F}$. Since secret shares are uniformly random in $\mathcal{F}$, the adversary cannot tell the difference between the two outputs. That is:

- $\Pr(f([\vec{D}], [\vec{\Lambda}]) \in \mathcal{F}) = \Pr(f([\vec{D}'], [\vec{\Lambda}']) \in \mathcal{F})$

This implies that $\epsilon = 0$ and f is 0-DP. In addition, since the layer function L^k is a parallel composition of independent calls of f, the output distributions of L^k on two neighboring datasets are the same. Therefore, L^k is also 0-DP. The same analysis is applicable for g and L^h, and we conclude L^h is 0-DP. Let p be the prediction function:

- $p([T_1], [\tau]) \rightarrow c_\tau$

where $[T_1]$ indicates the secret shares of T_1, a collection of outputs from each layer function L^k, L^h, and τ is a secretly shared user record. c_τ is the classification result only known to the user. Because the class labels are ϵ-DP, based on the robustness to post-processing property of DP, the prediction result is ϵ-DP, and so does T_1.

Another important consequence of the above proof is that since noise is only added at the leaf level, the total amount of noise added to the tree in our approach is significantly less than the existing solution. The empirical evaluation is presented in Section 4.

C.3 DP Mechanisms with MPC

We consider MPC for the Laplace mechanism and the Exponential mechanism. We want to secretly sample random numbers from appropriate distributions using basic MPC operations. Prior works have shown general methods of running a DP mechanism with a secret sharing-based scheme [31, 40]. Eigner et cl. described technical details of handling floating points when considering DP mechanisms in MPC [10]. We below summarize the basic approaches of constructing the two DP mechanisms using secret sharing-based MPC. We assume that the following basic MPC functionalities are available: addition add, multiplication mul, multiplication-by-constant $cmul$, exponentiation exp, a binary logarithm $log2$, random number generator between 0 and 1 $rand$, and comparison leq. We denote $[x]$ as a secret shared value of x.

C.3.1 Laplace Mechanism with MPC. Recall that the Laplace mechanism samples noise from the Laplace distribution $Lap(b)$ with a scale parameter $b = \Delta/\epsilon$, where Δ is a query sensitivity and ϵ is a privacy loss parameter. Algorithm 12 generates $Lap(b)$ in a secret share format and adds the secret noise to a secret share value of y. Namely, it performs $\tilde{y} = y + Lap(b)$ with MPC.

The idea is to sample a noise from the Laplace distribution using an inverse transform sampling with the exponential distribution. That is, the Laplace distribution can be viewed as the double exponential distribution, and it holds that $X_1 - X_2 \sim Lap(b)$ where X_1 and X_2 are independent and identically distributed $Exp(1/b)$. In addition, using an inverse transform sampling, we have $Exp(1/b) = -b \ln \mathcal{U}$, where $\mathcal{U}$ is a uniform random variable in $(0, 1)$. Thus, a random variable Y with distribution $Lap(b)$ can be created by:

$$Y = b \ln \mathcal{U}_1 - b \ln \mathcal{U}_2$$

where $\mathcal{U}_1$ and $\mathcal{U}_2$ are independent and identically distributed uniform distribution on $(0, 1)$.

Algorithm 12 MPC-LM($[y], \epsilon, \Delta$)

Require: Secret value $[y]$, privacy parameter ϵ, sensitivity Δ

```
b ← Δ/ε
for i ∈ {1, 2} do
    [u_i] = MPC-Rand()
    [r_i] = MPC-Log2([u_i])
    [r_i] = MPC-Cmul([r_i], b ln 2)
[r] = MPC-Add([r_1], −[r_2])
[ỹ] = MPC-Add([y], [r])
return [ỹ]
```

C.3.2 Exponential Mechanism with MPC. Algorithm 13 runs the Exponential mechanism with MPC when given secret-shared quality scores of d candidate items. Similar to the Laplace mechanism, we want to implement the Exponential mechanism by only using the basic functionalities that MPC supports. The idea is as follows. Given a set of d items $\{a_1, \ldots, a_d\}$ and a quality function q, we first compute probabilities $p_{a_i} = \frac{exp(\epsilon q(D, a_i)/2\Delta_q)}{\sum_i^d (exp(\epsilon q(D, a_i)/2\Delta_q))}$ for each a_i. Then, we can compute d intervals of the cumulative probabilities $\{p_1, \ldots, p_d\}$ where $p_1 = p_{a_1}$ and $p_i = p_{i-1} + p_{a_i}$ for $2 \le i \le d$. By sampling a uniform random number between 0 and 1 from $\mathcal{U}$, we return item a_i if the random number falls into the i-th interval, i.e., $p_{i-1} < \mathcal{U} \le p_i$.

In Algorithm 13, there is an option whether we return a selected index as a plaintext or a secret-shared value. If the index cannot be revealed, we need to linearly scan all the items (Line 9-16)[40]. Otherwise, we can perform an efficient binary search to find the index [10].

Algorithm 13 MPC-EM($\{[q_1], \ldots, [q_d]\}, \epsilon, \Delta$)

Require: Set of secret scores $\{[q_1], \ldots, [q_d]\}$, privacy parameter ϵ, sensitivity Δ

```
[p_0] = [0]
for i ∈ {1, ..., d} do
    [s] = MPC-Cmul([q_i], ε/2Δ)
    [r] = MPC-Exp([s])
    [p_i] = MPC-Add([p_{i−1}], [r])
[u] = MPC-Rand()
[u] = MPC-Mul([u], [p_d])
[index] = [0]
for i ∈ {1, ..., d} do
    [f] ← MPC-Leq([u], [q_i])
    [index] ← [f][i] + (1 − [f])[index]
return [index]
```

C.4 Implementation Details

In addition to the information provided in Section 4, here we provide additional implementation details. We implemented two different DP methods for leaf updates, one is based on the Laplace mechanism and the other is based on the Exponential mechanism. If not mentioned explicitly, the Laplace mechanism is used for DP leaf updates since we observed that the Laplace mechanism tends to offer a better accuracy than the Exponential mechanism, which we will show in our experiments. In addition, when implementing CDP+PMPC-ID3, we use the Exponential mechanism to select the best attribute since it is efficient and commonly adopted in practice. The MPC-ID3 protocol works over integers where we use 32-bits for the size of secret share. For CDP+PMPC-ID3 and ODP+FMPC-ID3, to handle real numbers needed for the DP mechanisms, fixed point numbers are used where the size of secret share is set to 32-bits for the Titanic and Heart datasets and 50-bits for the Adult dataset.

Dataset	Sample size	# of attributes	Domain size
Titanic	887	6	5
Heart	297	14	5
Adult	32651	13	7

Table 1: Datasets

C.4.1 Implementation of MPC-based DP Mechanisms. We implemented MPC-based Laplace mechanism and the Exponential mechanism using MPyC. For the Laplace mechanism, we followed the implementation of Algorithm 12 by Thissen [37]. For the Exponential mechanism, we referred to Table 4 by Eigner et. al [10] and the algorithm descriptions in Section 8.2 by Wu [40]. When searching a selected item, we perform an oblivious search by linearly scanning items since the selected index needs to be hidden instead of performing a binary search as in [10]. In addition, we use the common exp-normalize trick to prevent overflow where the maximum utility (i.e., the maximum quality score among the q_is used in the MPC-Exp sub-protocol of Algorithm 13 in Appendix C.3.2) is subtracted from the quality scores. This trick is also used in the Diffprivlib library [17]. To handle exponent underflows, for large negative numbers, we obliviously clip the values to return zeros. In our experiments, we set the exponent threshold to -5. That is, we set the output of the function to 0 for any exponent below -5. This did not affect model performance too much. Note that although we can increase the secret share size to handle overflow and underflow issues, this can easily result in computational infeasibility due to the expensive cost of secure exponentiation operations.

Using MPyC, we also implemented basic arithmetic operations required for the DP mechanisms, including the secure evaluation of an exponential function and a logarithmic function, following the algorithms presented by Thissen [37]. For accuracy evaluation that does not require MPC, we use the Laplace mechanism and the Exponential mechanism from the Diffprivlib library [17].

D EXTENSION TO RANDOM FOREST

In this section, we extend our MPC+DP decision tree framework to random forest. The recent works on a DP random forest implement a DP version of [12, 17] where a split attribute is randomly chosen. (We term this specific algorithm as extra trees.) Thus, DP noise is only introduced at leaf nodes. However, unlike its non-DP version, each tree trains on a disjoint subset of the data (i.e., bootstrap without replacement), which allows us to use the parallel composition. That is, as long as each tree is ϵ-DP, so is the random forest. On the other hand, implementing a DP version of random forest in the centralized setting is not privacy budget-efficient compared to the above DP extra trees because the best split needs to be determined under DP. By introducing MPC, our solution can provide a DP random forest without spending a privacy budget on split nodes.

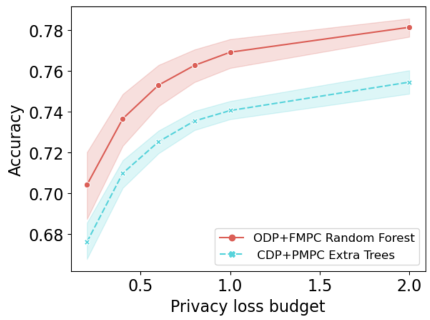
We implement the extra trees-based DP random forest and ID3-based DP random forest with our solution. Figure 6 shows the relationship between accuracy and privacy loss ϵ. For each approach, we varied the maximum depth from 2 to 5 and the number of trees from 1 to 10. Note that increasing the number of trees do not necessarily help to improve the accuracy especially for small datasets since each tree gets trained on a very small number of samples. As seen in the plots, we observed that our solution provides a better accuracy. As mentioned before, in our current implementation, we adopt sampling without replacement. Next we discuss how to choose the parameters when using sampling with replacement.

D.1 Parameter Selection and Privacy Amplification

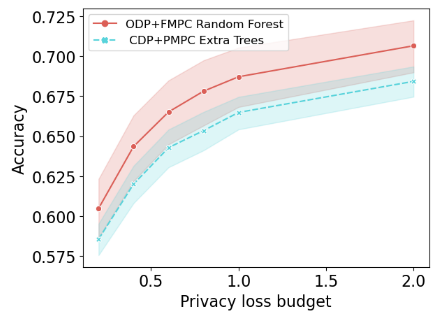
Here we consider how to distribute the privacy budget ϵ among the trees in the forest and how the number of trees affects the selection of ϵ. First, we prove the following claim:

Claim D.1. *Let γ be a sampling rate in (0,1) and κ be the number of trees in a random forest. If each tree is built using ODP-FMPC-ID3 on a subset of data randomly generated with Poisson sub-sampling with rate γ and privacy budget ϵ, then the resulting random forest achieves $\kappa\gamma\epsilon$-DP in the worst case.*

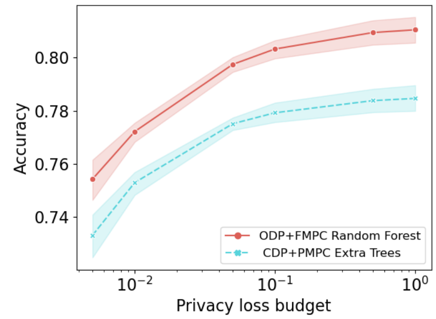
To prove the claim, we adopt Theorem 8 given in [2] which states that if a mechanism $\mathcal{M}$ is ϵ-DP, it achieves at least $\gamma\epsilon$-DP when applying to a subset of data sampled with Poisson sub-sampling with sampling rate γ. Thus, each tree produced by ODP-FMPC-ID3 achieves $\gamma\epsilon$-DP. If we treat each tree is produced sequentially, then



(a) Titanic



(b) Heart



(c) Adult

Figure 6: Accuracy vs. privacy loss for random forest, varying the max depth from 2 to 5 and the # of trees from 1 to 10

the random forest is generated by a sequence of $\gamma\epsilon$-DP mechanisms. Applying the sequential composition theorem, the random forest achieves $\kappa\gamma\epsilon$-DP. This is the worse case since the sequential composition theorem assumes the mechanisms apply to the same dataset. However, in our case, the sub-datasets are randomly sampled. Based on the claim, to achieve ϵ-DP, we can set $\gamma = 1/\kappa$.