Toward Efficient and End-to-end Privacy-preserving Distributed Gradient Boosting Decision Trees

Shuai Yuan*, Hongwei Li(Corresponding author)*, Xinyuan Qian*, Meng Hao*, Yixiao Zhai* and Xiaolong Cui†
* School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
† School of Computer & Communication Engineering, University of Science and Technology Beijing

***Abstract*—Gradient Boosting Decision Trees (GBDTs) are popular machine learning models due to its simplicity, effectiveness, and interpretability. Recently, to alleviate serious privacy leakages in conventional centralized methods, researchers have proposed several privacy-preserving distributed GBDT solutions. However, those approaches still suffer from either insufficient privacy protection or significant runtime and communication overhead. In this paper, we propose an efficient and end-to-end privacy-preserving distributed GBDT framework, called PPD-GBDT, which uses differential privacy, polynomial approximation, and fully homomorphic encryption to achieve comprehensive privacy protection. Specifically, during the boosting phase, we design a novel model preparation method to improve the efficiency of prediction with acceptably slight accuracy/RMSE loss while preventing data owners' corruption. On the other hand, for the prediction phase, we propose a customized secure prediction method, which effectively prevents the malicious server from stealing private information. Besides, we conduct extensive experiments on six datasets and compare with three prior schemes. Evaluation results show that our privacy-preserving scheme achieves lower runtime and up to 40× less communication overhead compared to the state-of-the-arts.**

***Index Terms*—Privacy-preserving machine learning, Gradient Boosting Decision Trees, Homomorphic Encryption.**

I. Introduction

Gradient Boosting Decision Trees (GBDTs) have widely used in data mining and scientific research [1], due to its high accuracy, stability, and interpretability. Recently, Machine Learning as a Service (MLaaS) employed by many tech companies like Google and Microsoft is highly popular to provide automatic training and prediction services. As shown in Fig. 1, we consider a typical MLaaS scenario, where the server wants to build GBDTs using data provided by the data owner, and provides prediction services for users. However, this paradigm leads to serious privacy issues, such as leaking the sensitive training data, query samples, and prediction results.

To mitigate such problems, several works have explored designing privacy-preserving decision tree methods. On the one hand, Akavia et al. [2] and Fang et al. [3] proposed fully homomorphic encryption (FHE) based model evaluation, in which the data owners and users send encrypted data to the server for both training and prediction. Despite desirable privacy guarantees, these methods result in significant runtime and communication overhead, such as 20.3 hours for training only a tree of depth 4. On the other hand, Li et al. [4] studied a practical federated environment with relaxed privacy

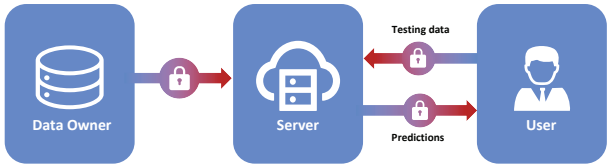


Fig. 1: An example of a MLaaS scenario.

constraints, but the scheme cannot be implemented if the user's instance lacks the ID attribute. Liu et al. [5] proposed a federated extreme gradient boosting scheme supporting forced aggregation by combining the advantages of secret sharing and homomorphic encryption. However, this scheme only considers the training phase, and there lacks a secure design for the prediction phase. Aminifar et al. [6] proposed a privacy-preserving distributed extremely randomized trees (ERT) approach for privacy-preserving utilization of the ERT algorithm in a distributed setting. Each user can get the aggregated results during the training phase, which is not secure. In general, these cryptographic approaches did not provide comprehensive privacy protection for both boosting and prediction phases in terms of private data, intermediate values, and prediction results. Therefore, it is urgent to design an efficient decision tree scheme that enables comprehensive privacy conservation for both training and prediction phases with low overhead.

To address the above issues, we propose an end-to-end privacy-preserving distributed GBDT framework, called PPD-GBDT, which exploits differential privacy (DP), polynomial approximation (PA), and fully homomorphic encryption (FHE) techniques. The PPD-GBDT effectively provides comprehensive privacy guarantees for both boosting and prediction phases. Specifically, during the boosting phase, data owners train DP-protected GBDTs to hide the private information of training data. Afterward, the GBDTs are converted into polynomial approximation trees (PATs) by our PA method to boost efficiency of the prediction phase. Compared to FHE-based alternatives that train GBDT over encrypted samples, our method achieves significant efficiency improvement with provably quantifiable privacy guarantees. On the other hand, for the prediction phase, we design a secure prediction method to efficiently implement expensive comparison operations without sacrificing model performance, thereby combining FHE to obtain prediction results on PATs. The testing data

ICC 2023 - IEEE International Conference on Communications | 978-1-5386-7462-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICC45041.2023.10279189

and prediction results can be hidden from the cloud server at this stage. We summarize our contributions as follows:

- We propose an efficient and end-to-end distributed GBDT framework (PPD-GBDT) that provides comprehensive privacy protection for both boosting and prediction phases.
- We design a novel model preparation method for the boosting phase to improve the efficiency of prediction while preventing model owners' corruption. For the prediction phase, we propose a customized secure prediction method to prevent the malicious server from stealing private information.
- We conduct extensive experiments on six datasets. Compared to state-of-the-art schemes, our privacy-preserving approach achieves up to 6× less runtime and 40× less communication overhead with slight accuracy/RMSE loss.

The remaining parts of this paper are organized as follows. In Section II, some background knowledge is briefly introduced. In Section III, we describe the details of our scheme. In Section IV, we conduct extensive experiments to evaluate the performance. Finally, Section V concludes the paper.

II. Preliminaries

A. Gradient Boosting Decision Trees

The GBDT is an ensemble ML algorithm training decision trees sequentially. The algorithm consists of two main steps: training decision trees one by one, with each tree attempting to reduce the residuals of previous trees; and then summing the outputs of all trees as the final result.

Given a training dataset $T = \{(x_1, y_1), ..., (x_N, y_N)\}$ where $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathcal{Y} \subseteq \mathbb{R}$, and a certain loss function $\mathcal{L}(y, f(x))$, the goal of GBDT is to find an estimation function $\hat{f}(x)$ that maps every vector x_i to label y_i to minimize the expected value of the loss function $\mathcal{L}$. The essence of boosting is to approximate the residual $\overline{y}_i$ by the negative gradient of the loss function, i.e., $-[\frac{\partial \mathcal{L}(y, f(x_i))}{\partial f(x_i)}]_{f(x)=f_{m-1}(x)}$. We can regard the training of each ordered weak learner as an iteration. In each iteration, we first estimate the residual using the above loss function, then use the approximated residual to train the decision tree. In the prediction phase, the GBDT sums up the predicted values of all trees as the final prediction.

B. Differential Privacy

Differential privacy proposed by Dwork et al. [7] is a wildly used approach to prevent data leakage in ML. The central idea is to obfuscate personal information when an adversary attempts to obtain it from databases. As a result, an adversary cannot infer sensitive information at the individual level from normal query results. Here is the formal definition:

***Definition** 1:* (ε, δ)-Differential Privacy. Let ε be a privacy budget, δ be a failure probability, and f be a randomized function. The function f is said to provide (ε, δ)-Differential Privacy, if for any two datasets D and D' that differ in a single record and any output O of function f,

$$Pr[f(D) \in O] \leq e^{\epsilon} \cdot Pr[f(D') \in O] + \delta.$$

Here ε is a positive real number. (ε, δ)-Differential privacy is usually achieved by adding noise calibrated to the sensitivity of a function using two main methods, Laplace Mechanism and Exponential Mechanism.

C. Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is a public-key cryptosystem that allows computing and evaluating an arithmetic circuit over encrypted data. The standard definition for FHE [8] is given as follows:

***Definition** 2:* Let $\mathcal{M}$ be the message space, and λ be the security parameter. A public-key FHE scheme consists of four probabilistic polynomial time (PPT) algorithms $\mathsf{FHE} = (\mathsf{FHE.KeyGen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$.

- Key Generation: $(pk, evk, sk) \leftarrow \mathsf{FHE.KeyGen}(1^{\lambda})$, outputs a public encryption key pk, a evaluation key evk, and a secret decryption key sk.
- Encryption: $c \leftarrow \mathsf{FHE.Enc}(pk, \mu)$, using the public key pk, encrypts a message $\mu \in \mathcal{M}$ into a ciphertext c.
- Decryption: $\mu \leftarrow \mathsf{FHE.Dec}(sk, c)$, using the secret key sk, decrypts a ciphertext c to recover the message $\mu \in \mathcal{M}$.
- Homomorphic Evaluation: $\hat{c} \leftarrow \mathsf{FHE.Eval}(C, (c_1, \cdots, c_l), pk, evk)$, using the public key pk and the evaluation key evk, applies a circuit C: $\mathcal{M}^l \rightarrow \mathcal{M}$ to $[c_1], ..., [c_l]$ and outputs a ciphertext $\hat{c}$.

III. Methodology

A. Threat model

Our system architecture is illustrated in Fig. 2. In the proposed system, there are three kinds of entities: the data owner (DO), the cloud server (CS), and the querying user (QU). The DOs want to learn GBDTs without sharing local data with any entity in the framework. The CS collects models to predict encrypted data sent by the QU. We consider the following threat model:

- Malicious CS: Consistent with prior work [2], we assume that the CS is malicious and may follow any arbitrary attack strategy to get sensitive data.
- Honest-but-curious and colluding DOs: We assume that DOs may collude to try to acquire private information from other DOs.
- Honest-but-curious QU: Similar to most of the existing works on privacy-preserving machine learning, we assume that the QU correctly follows the algorithm and protocols, but may try to learn the secret model in the process.

We also assume that secure channels are used in all communications, thus, man-in-the-middle and trivial snooping attacks are prevented. Moreover, we believe that there is no corruption between different entities. Based on the threat model above, our PPD-GBDT can ensure that (i) the malicious CS cannot learn additional information except for the expected output and polynomial approximation trees, and (ii) the colluding DOs cannot get the parameters of other honest DOs.

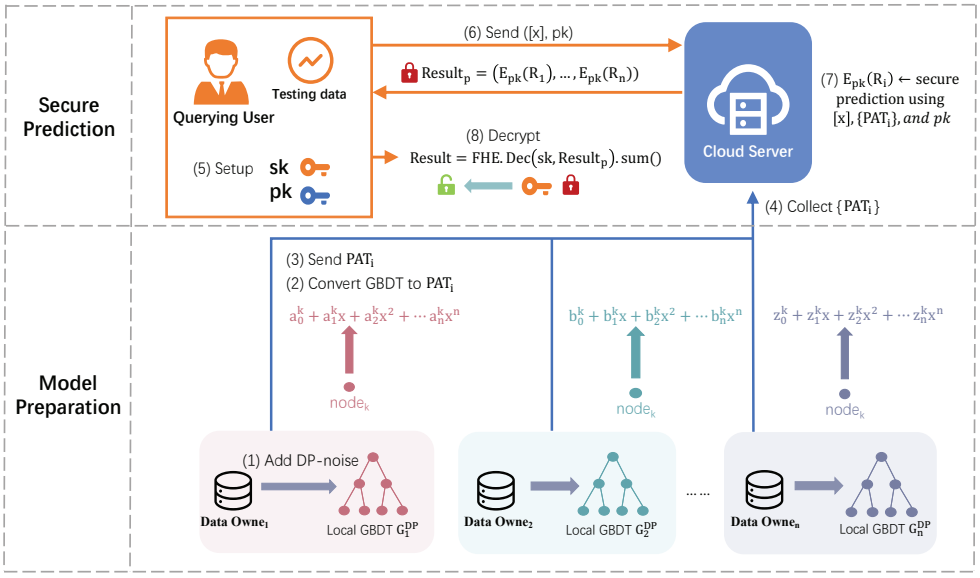


Fig. 2: System Overview.

B. General Idea

In many distributed works based on tree models, encrypted data needs to be uploaded during the training phase to protect privacy, or techniques such as secure multi-party computation are used, which unfortunately result in large amounts of time and communication overhead. Many efforts do not implement comprehensive privacy protection and fail to defend against malicious servers. To solve these problems, we propose an end-to-end privacy-preserving distributed GBDT framework, called PPD-GBDT.

From a high-level perspective, our scheme consists of two phases: (1) model preparation; (2) secure prediction. The purpose of the model preparation is to protect the local data of DOs and to improve the efficiency of the model prediction. Every DO trains the GBDT on local data and performs distinct transformations on GBDTs. The secure prediction phase refers to the process of securely obtaining the models' outputs. The CS implements the customized secure prediction method to get the outputs of all models with encrypted testing data. The predictions are sent to the QU, who decrypts and sums them to get the true results. We present the details of each phase as follows.

C. Model Preparation

Firstly, we need to protect the DOs' local data during the boosting phase. Many schemes choose to encrypt the local data and train the model on the server side, but this places a huge burden on the server and causes significant communication overhead. To address this issue, we design the **model preparation** method to improve the efficiency of prediction while keeping training data secure. Specifically, we decide to have DOs train models locally and convert the DP-protected models to polynomial approximation trees (PATs) which are shared to CS instead of transmitting encrypted datasets.

Inspired by DPBoost [9], we add DP noises during the training process to protect the node privacy of each decision tree. Because the runtime of GBDTs is prohibitively high when predicting under ciphertext, DOs cannot upload the DP-protected GBDTs directly to the CS. The main reason for this is the comparison function on non-leaf nodes. In many distributed efforts, the processing of comparison functions is done using secure multi-party computation, but this not only introduces extra non-colluding servers but also leads to significant additional overhead. Thus we explore converting the comparison function to crypto-friendly polynomials.

First of all, we assume that the data are normalized to the interval $[-1, 1]$. Here, we consider the comparison function $I_0 : \mathbb{R} \to \{0, 1\}$ with threshold zero, defined by: $I_0(x) = 1$ if $x \geq 0$ and $I_0(x) = 0$ otherwise. The polynomial function would be the solution to the following optimization problem:

$$\phi = \min_{p \in P_n} \int_{-2}^{2} (I_0(x) - p(x))^2 dx,$$

where P_n is the set of polynomial functions of degree at most n over the reals. We assume that the polynomial approximation function for $I_0(x)$ is as follows:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + ... + \beta_{n-1} x^n$$

where β_i is the coefficient of the polynomial, and n is the degree of the polynomial. For m samples, we can write m of the above system of linear equations ($m \geq n$). So the final matrix expression can be formalized as:

$$\phi = \min \|X\beta - Y\|_2^2 ; X \in R^{m \times n}, \beta \in R^{n \times 1}, Y \in R^{m \times 1}$$

where X is the sample matrix, β is the parameter matrix, and Y represents the true value. The solution of the above equation

leads to:

$$\|X\beta - Y\|_2^2 = \beta^T X^T X\beta - 2\beta^T X^T Y + Y^T Y$$

Then, use the above equation to derive the vector β:

$$\frac{\partial(\|X\beta - Y\|_2^2)}{\partial\beta} = 2(X^T X\beta - A^T Y)$$

Let the result of the above equation equal 0, and we get:

$$\beta = (X^T X)^{-1} X^T Y$$

The above equation is the analytical solution of the minimum mean-square method, which is a globally optimal solution. We have used L2 regularization to turn the analytic solution into the following form:

$$\beta = (X^T X + \lambda E)^{-1} X^T Y$$

where λ is the canonical parameter, and E is an $n \times n$-dimensional identity matrix. Finally, we obtain the coefficients of the approximate polynomial. We use the coefficients as an attribute of the node and remove the threshold information of the node. By applying the low degree polynomial approximation to non-leaf nodes, we can convert GBDTs into PATs, which is highly useful in the subsequent secure prediction approach.

Algorithm 1 Secure Prediction

Input: The testing data x; the degree n; public key pk; secret key sk

Output: The encrypted result enc_result

function $Enc(x, pk)$
  $[x] \leftarrow \mathsf{FHE.Enc}(pk, x)$
  return ($[x]$, pk)
function $PAT\_Predict(node, [x], pk)$
  if $node$ is a leaf **then**
    return $node.leaf\_value$
  $result \leftarrow$ the approximate coefficients of $node$ and $[x]$ do polynomial operations
  $l\_res \leftarrow PAT\_Predict(node.left, [x], pk)$
  $r\_res \leftarrow PAT\_Predict(node.right, [x], pk)$
  return $result \times l\_res$ + (1 - $result$) $\times$ r_res
$enc\_result \leftarrow [\,]$
for each PAT_i in DO **do**
  result $\leftarrow PAT\_Predict(PAT_i.root, [x], pk)$)
  enc_result.append(result)

D. Secure Prediction

In our PPD-GBDT, after these DOs upload PATs, the CS can predict the results for testing data. The PATs can effectively combine with FHE and implement the comparison function of nodes. However, for the prediction process, we need to determine the true prediction path, so we design the customized **secure prediction** method. Since we assume that the CS is malicious, the QU cannot send plaintext directly to the CS. As we mentioned in Section II-C, the QU needs to generate the public/private key pair (pk, sk) using $\mathsf{FHE.KeyGen}(1^\lambda)$, encrypts testing data using $Enc(x, pk)$ in Algorithm 1, and finally sends $([x], pk)$ to the CS. In our PPD-GBDT, after sending his/her query, the QU can be offline until receiving the encrypted classification results. Currently, the CS does not have access to the local data used by the DOs for training and QU's testing data for prediction.

As shown in Algorithm 1, the CS receives the PATs and encrypted testing data. The $node.left$ and $node.right$ represent its left child and right child nodes, respectively, and the $PAT_i.root$ is the root node. For each PAT_i, $PAT\_Predict(PAT_i.root, [x], pk)$ is used to get the predicted result. If the current node is a leaf node, return the leaf value, otherwise use the polynomial prediction circuit under the ciphertext to get the result. The $result$ requires the parameter $node.factor$ of this node polynomial, the ciphertext enc_data corresponding to the node features, and the public key pk as input. It obtains the encrypted result of computing $a_0 + a_1 \times enc\_data + ... + a_n \times enc\_data^n$ with FHE, where $(a_0, a_1, ..., a_n)$ is the coefficients of the $node$ after polynomial approximation. Because the input data is encrypted, the polynomial prediction result is in the form of ciphertext (approximately equal to 1 or 0 after decryption).

From the root node, we multiply the $result$ with the ciphertext going to the left subtree l_res by recursively calling $PAT\_Predict$, and then add $(1 - result)$ with the ciphertext result going to the right subtree r_res. When the last leaf node is reached, the ciphertext is multiplied by the value of the leaf node. By doing this above, only the predicted valid paths are retained during decryption, and the rest of the invalid paths are excluded because the decrypted values of the intermediate nodes are close to zero. Finally, we get the prediction results for all PATs. The output of each node is an encrypted value, so the CS does not get any intermediate results.

Algorithm 2 The PPD-GBDT framework

Input: The testing data x; the degree n; public key pk; secret key sk

Output: The predicted result $final\_result$

for each DO_i **do**
  train GBDT with DP on local dataset
  convert decision trees in GBDT to PATs (degree n)
QU **executes:**
  $[x] \leftarrow Enc(x, pk)$
  send $[x]$ to the CS
CS **executes:**
  $total\_result \leftarrow [\,]$
  for each DO_i **do**
    result $\leftarrow$ **secure prediction** using $[x]$, PATs, pk
    $total\_result$.append(result)
  send $total\_result$ to the QU
QU **executes:**
  for each result in $total\_result$ **do**
    result $\leftarrow \mathsf{FHE.Dec}(sk$,result)
  $final\_result \leftarrow total\_result.sum()$

E. Putting It All Together

We show the complete PPD-GBDT framework in Algorithm 2. At the beginning, each DO trains GBDT with DP using the local dataset. After the training, the DOs need to convert the GBDTs to PATs and submit them to the CS.

The testing data is encrypted by the locally-generated public key of QU and then sent to the CS. The ciphertext $[x]$ is used as input to make a secure prediction on the PAT (i.e., $PAT\_Predict()$) using FHE. After getting all encrypted predictions, the CS sends them to the QU, who decrypts the ciphertext and sums them up as the final prediction.

Each DO trains GBDT locally, without sharing local data with any entity and having any interaction with each other. Therefore, there is no access to any other valuable information for colluding DOs. If the CS is malicious, not only does it not have access to the training data of DOs, it also does not know the testing data. Our secure prediction method protects the intermediate results and predictions. So this can effectively protect the privacy of DOs and QUs against privacy attacks by a malicious CS.

IV. Evaluation and Results

A. Experimental setup

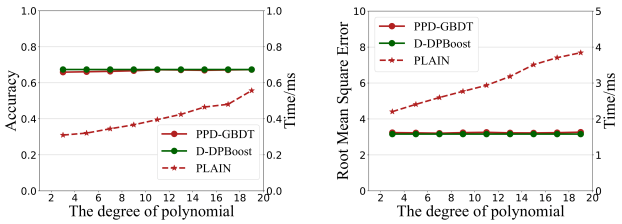
To benchmark the effectiveness and efficiency of our scheme, we compare **PPD-GBDT** with four approaches: 1) **PPCP**: Giacomelli et al. [10] proposed the distributed privacy-preserving framework using linearly-homomorphic encryption; 2) **PPDE**: Liu et al. [11] proposed the privacy-preserving decision tree evaluation scheme, which combines additively homomorphic cryptosystem and additive secret sharing. 3) **D-DPBoost**: We implemented the DPBoost [9] and extended it to distributed scenarios to test the fluctuations of accuracy or RMSE; 4) **PLAIN**: This work refers to the secure prediction of our scheme in plaintext after the DOs have uploaded PATs and can help us test the time overhead of secure prediction.

We implement and evaluate our scheme to demonstrate its practicality. All experiments are performed on an Intel Core i7 CPU@2.2GHz, ignoring network latency. The implementation is in Python, using the DPBoost library [9] for differential privacy, and the TenSEAL library [12] for fully homomorphic encryption [13]. We use six datasets from the UCI machine learning repository [14]. The settings of the various parameters are indicated in the different experimental contents.

B. Efficacy

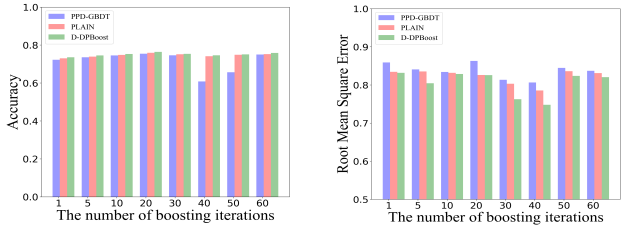
Accuracy/RMSE Comparison with Prior Works. We compare with other schemes according to the number of boosting iterations, as shown in Fig. 4. It can be seen that PPD-GBDT reduces the accuracy by less than 0.01 and increases the RMSE by less than 0.04 compared with PLAIN. Both PPD-GBDT and PLAIN do not fluctuate much with the number of iterations. The GBDTs are overfitted gradually when the number of iterations is greater than 40.

The Effect of Polynomial Degree. We study the effect of polynomial degree n on our scheme. We set up 10 DOs, each with 50 trees and 30 leaves per tree. As shown in Fig. 3, the



(a) Heart Disease (b) Abalone

Fig. 3: The impact of different degrees on the models with two datasets. The solid line represents the Accuracy or RMSE; the dashed line represents the runtime overhead.



(a) Adult (b) Wine Quality

Fig. 4: Accuracy/RMSE comparison with prior works.

results of PPD-GBDT are consistent with D-DPBoost, which indicates that converting a decision tree to a PAT brings few errors. The time overhead of PLAIN increases as the degree gets larger because the complexity of the computation required for each node also increases. Ultimately, we can conclude that the results do not change much as n increases. However, the time overhead becomes larger as the degree increase, so the ideal choice is to set the degree to 3.

C. Efficiency

Efficiency Comparison with Prior Works. We compare the efficiency of PPD-GBDT with previous works. Table I summarizes the comparison result. It is observed that compared with PPCP, our scheme has a significant advantage in terms of communication overhead (40× less). This is because PPCP needs to encrypt the tree before sending it to the server, which results in extremely large ciphertext trees in transmission. For PPDE, it has a larger overhead than PPD-GBDT because of its used heavy cryptographic building

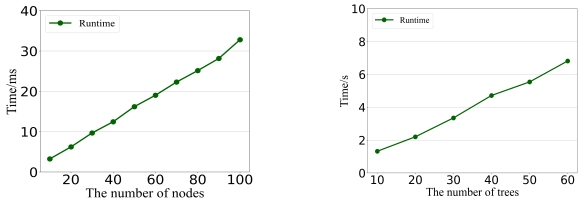
Table I: Efficiency comparison with prior works

Dataset	Scheme	RunTime			Communication
		Boosting	Prediction	Total	
Spambase	PPCP	1.01s	45.00s	46.01s	12.35MB
	Our	5.82s	27.82s	**33.64s**	**314KB (40×)**
Nursery	PPDE	-	-	66.29s	661.64KB
	Our	1.92s	8.35s	**10.27s**	**45KB (14 ×)**

blocks. Therefore, when applied to federated learning, PPD-GBDT is still more practical than PPDE.

Next, we will test the two phases in PPD-GBDT separately and discuss the experimental results on the effects of important parameters involved.

Efficiency of Model Preparation. First, we conduct experiments on the time overhead of model preparation. Similar as [9], we consider the time overhead of converting GBDTs to PATs. As shown in Fig. 5, the overhead of the conversion process is mainly related to two parameters, the number of nodes in a tree and the number of trees in GBDT. It is observed that for a single tree, the time overhead increases linearly as the number of nodes increases. We assume that the transformation of each tree is performed sequentially rather than in parallel. Therefore, as the number of trees increases, the time overhead becomes larger.



(a) The impact of nodes (b) The impact of trees

Fig. 5: Runtime overhead of converting GBDTs to PATs.

Second, we examine the communication overhead of the model preparation phase (i.e. the size of PATs). The transmission overhead of each DO is shown in Table II. The difference in storage size between PATs and LGBM is only 13KB, which is negligible. Therefore, our solution does not bring much additional overhead and is more friendly to edge devices with weak computation and storage capabilities.

Table II: Communication overhead for model preparation

Dataset	GBDT Type	Size
Adult	LGBM [15]	78KB
	DPBoost [9]	82KB
	PATs	91KB

Table III: Runtime overhead for secure prediction

Dataset	Max Depth	PLAIN	PPD-GBDT
Spambase	2	0.28s	0.49s
	3	0.58s	7.58s
	4	1.06s	8.35s
	5	1.85s	17.33s
	6	2.20s	19.37s

Efficiency of Secure Prediction. We study the runtime of secure prediction in Table III. Considering the practicalities of application cryptography, the server will have strong computational power so that secure prediction can be executed in parallel. When the maximum depth of GBDT is greater than 3, the time will increase substantially. This is mainly because our scheme needs to calculate the ciphertext of all nodes.

V. Conclusion and Future Work

In this paper, we construct a privacy-preserving framework for distributed gradient boosting decision trees, called PPD-GBDT. The architecture addresses two key challenges of state-of-the-arts, i.e., lack of comprehensive privacy protection, and huge overhead. Extensive evaluations on six datasets shows that PPD-GBDT can achieve privacy protection with low overhead at only a slight performance reduction. In the future, we will further investigate how to reduce the runtime overhead of ciphertext prediction and improve the performance of the model while ensuring privacy.

Acknowledgment

This work is supported by the National Key R&D Program of China under Grant 2022YFB3103500, the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101360001, the National Natural Science Foundation of China under Grants 62020106013 and 61972454, the Sichuan Science and Technology Program under Grants 2020JDTD0007, the Fundamental Research Funds for Chinese Central Universities under Grant ZYGX2020ZB027.

References

[1] W. Fang, J. Zhou, X. Li, and K. Q. Zhu, "Unpack local model interpretation for gbdt," in *Proceedings of DASFAA*, 2018, pp. 764–775.
[2] A. Akavia, M. Leibovich, Y. S. Resheff, R. Ron, M. Shahar, and M. Vald, "Privacy-preserving decision tree training and prediction against malicious server," *Cryptology ePrint Archive*, 2019.
[3] W. Fang, C. Chen, J. Tan, C. Yu, Y. Lu, L. Wang, L. Wang, J. Zhou *et al.*, "A hybrid-domain framework for secure gradient tree boosting," *arXiv preprint arXiv:2005.08479*, 2020.
[4] Q. Li, Z. Wen, and B. He, "Practical federated gradient boosting decision trees," in *Proceedings of AAAI*, vol. 34, no. 04, 2020, pp. 4642–4649.
[5] Y. Liu, Z. Ma, X. Liu, S. Ma, S. Nepal, R. H. Deng, and K. Ren, "Boosting privately: Federated extreme gradient boosting for mobile crowdsensing," in *Proceedings of ICDCS*, 2020, pp. 1–11.
[6] A. Aminifar, F. Rabbi, K. I. Pun, and Y. Lamo, "Privacy preserving distributed extremely randomized trees," in *Proceedings of SAC*, 2021, pp. 1102–1105.
[7] C. Dwork and J. Lei, "Differential privacy and robust statistics," in *Proceedings of STOC*, 2009, pp. 371–380.
[8] D. Benarroch, Z. Brakerski, and T. Lepoint, "Fhe over the integers: Decomposed and batched in the post-quantum regime," in *IACR International Workshop on Public Key Cryptography*. Springer, 2017, pp. 271–301.
[9] Q. Li, Z. Wu, Z. Wen, and B. He, "Privacy-preserving gradient boosting decision trees," in *Proceedings of AAAI*, vol. 34, no. 01, 2020, pp. 784–791.
[10] I. Giacomelli, S. Jha, R. Kleiman, D. Page, and K. Yoon, "Privacy-preserving collaborative prediction using random forests," *AMIA summits on translational science proceedings*, vol. 2019, p. 248, 2019.
[11] L. Liu, R. Chen, X. Liu, J. Su, and L. Qiao, "Towards practical privacy-preserving decision tree training and evaluation in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 2914–2929, 2020.
[12] A. Benaissa, B. Retiat, B. Cebere, and A. E. Belfedhal, "Tenseal: A library for encrypted tensor operations using homomorphic encryption," *arXiv preprint arXiv:2104.03152*, 2021.
[13] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Proceedings of ASIACRYPT*, 2017, pp. 409–437.
[14] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.
[15] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.