This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2024.3395928

Efficient and Privacy-preserving Outsourcing of Gradient Boosting Decision Tree Inference

Shuai Yuan, *Student Member, IEEE*, Hongwei Li (Corresponding Author), *Fellow, IEEE*, Xinyuan Qian, *Student Member, IEEE*, Meng Hao, *Student Member, IEEE*, Yixiao Zhai, *Student Member, IEEE*, Guowen Xu, *Member, IEEE*

***Abstract*—Recently, outsourcing machine learning inference services to the cloud has become increasingly popular. The inference process, however, remains an open question on *how to effectively protect the model owner's proprietary model, the user's sensitive data, and prediction results*. In this work, we propose an efficient and comprehensive privacy-preserving framework for outsourcing Gradient Boosting Decision Tree (GBDT) inference utilizing pseudorandom function and additively homomorphic encryption. Specifically, we first design a transformation method for GBDT to protect the node and structure privacy of the owner's model. On top of the protected model, we further propose customized comparison and random trees permutation protocols, which substantially boost the computation and reduce the communication cost of the outsourcing inference, while preventing the user from inferring privacy associated with GBDT. Besides, we provide rigorous security analysis, and extensive experiments on 7 real-world datasets and various models demonstrating that our scheme achieves up to 36 times less runtime and 69 times less communication compared to the state-of-the-arts.**

***Index Terms*—Privacy-preserving Machine Learning, Outsourcing Inference, Gradient Boosting Decision Tree, Additively Homomorphic Encryption.**

I. Introduction

Gradient Boosting Decision Tree (GBDT) has become a popular type of machine learning (ML) algorithm because of its efficiency, accuracy, and interpretability. With the development of efficient GBDT libraries [1] [2] [3], it can be widely applied in both data science competitions and industrial scenarios such as credit modeling [4], fraud detection [5], and medical diagnosis [6]. In short, GBDT builds many decision trees one by one, where each tree tries to reduce the residual of previous trees. In the prediction phase, the final result is obtained by summing up the outputs of all trees. With the widespread adoption of cloud computing, outsourcing GBDT inference services to the cloud is gaining more and more popularity.

However, this practice can lead to critical privacy issues. Firstly, the model owner's model is usually exclusive since training an effective model takes a significant investment in datasets, computational resources, and labor costs. The model owner would naturally not want to expose their models to the cloud server in plaintext. Secondly, the users' query data may be sensitive, such as financial information or medical data. Sending data directly in plaintext may easily compromise the user's privacy. Thirdly, the user might not like the server to be aware of the true predictions, such as financial judgments or medical diagnoses. Meanwhile, the user cannot infer information about GBDT based on predictions. The server should only be able to make predictions but do not know the real results.

Extensive efforts have been dedicated to enhancing the security of decision tree evaluation [7] [8] [9] [10]. However, these schemes grapple with the challenge of effectively balancing accuracy and overhead. In the realm of privacy-preserving decision tree evaluation (PPDE), research can be broadly categorized into two distinct approaches: perturbation-based and cryptography-based methods. Firstly, perturbation-based approaches [11] [12] [13] [14] predominantly involve the introduction of randomness or noise into the original database. These methods exhibit high efficiency without entailing time-consuming operations. However, the introduction of noise can significantly undermine model accuracy. Secondly, cryptography-based methods [15] [16] [17] ensure both high model accuracy and robust security. However, many of these methods [18] [19] [20] rely on resource-intensive cryptographic tools, resulting in substantial overhead that limits their practical applicability. Furthermore, it is important to note that many existing PPDE methodologies [8], [10], [21], [22], [23], [24], [25] fail to effectively scale to GBDT. This limitation arises from the fact that users, with access to predictions for each tree, can pinpoint sparse trees that reach leaf nodes "too early" [26]. A more detailed exploration of related work is available in Section II.

In this work, our focus lies in employing cryptographic methods to ensure the utmost accuracy of the GBDT while minimizing any associated overhead. To tackle the aforementioned challenges, we present an innovative secure framework. This framework empowers GBDT owners to deploy inference services within the cloud for their clients while upholding the confidentiality of proprietary models, sensitive user data, and prediction outcomes. Firstly, to protect the model owner's proprietary GBDT, we consider how to properly encrypt the decision trees in GBDT so that they can still function well during inference in the cloud. Note that a decision tree contains not only node privacy but also structural information, like the evaluation path over the feature vector, which also demands protection. Our solution involves encoding all decision trees in GBDT with pseudorandom function and additively homomorphic encryption. Secondly, to be compatible with the working paradigm of additively homomorphic encryption as well as protect users' sensitive data and prediction results, we design a secure comparison protocol. This protocol efficiently facilitates the comparison of internal nodes during prediction without leaking feature or threshold specifics. Importantly, the comparison process circumvents the need to traverse through all internal nodes, resulting in minimal overhead. Finally, to

TABLE I: Comparative summary of different schemes

Scheme	Techniques	Round	Leakage	Privacy		One server	One path	Support GBDT
				Node	Structure			
Bost et al. [7]	leveled-FHE	≥ 6	γ	✓	✗	✓	✗	✓
Wu et al. [27]	HE, OT	6	γ, d	✓	✓	✓	✗	✓
Cock et al. [8]	SS, OT	11 ($d \leq 9$)	γ, d	✓	✓	✗	✗	✗
Tai et al. [22]	AHE	4	γ	✓	✗	✓	✗	✗
Joye et al. [16]	AHE, OT	≥ 6	d	✓	✓	✓	✗	✓
Tueno et al. [23]	OT, GC	$4d$	d	✓	✗	✓	✓	✗
Tueno et al. [23]	ORAM	$d^2 + 3d$	d	✓	✗	✓	✓	✗
Kiss et al. [9]	AHE, OT, GC	4	γ	✓	✗	✓	✗	✓
Zheng et al. [18]	SS	$O(2^d)$	d, l	✓	✗	✗	✗	✓
Ma et al. [19]	SS, OT, GC	$2d - 1$	γ, d	✓	✓	✗	✓	✓
Bai et al. [20]	AHE, SS, OT	$8d$	γ, d	✓	✓	✗	✓	✗
Bai et al. [20]	SS, OT, PRF	$(3r_f + 5)d$	γ, d	✓	✓	✗	✓	✗
Ours	AHE, PRF	$2d$	γ, d	✓	✓	✓	✓	✓

Node Privacy: information on internal nodes and leaf nodes, **Structure Privacy**: tree structure and prediction paths, **One path**: traverse only the nodes on the predicted path, $\boldsymbol{\gamma}$: the number of decision nodes, $\boldsymbol{d}$: the tree depth, $\boldsymbol{l}$: bit length of each element in the feature vector, $\boldsymbol{r_f}$: rounds for evaluating PRF f, **leveled-FHE**: leveled Fully Homomorphic Encryption, **HE**: Homomorphic Encryption, **AHE**: Additively Homomorphic Encryption, **OT**: Oblivious Transfer, **SS**: Secret Sharing, **GC**: Garbled Circuit, **ORAM**: Oblivious Random-Access Memory, **PRF**: Pseudorandom Function.

prevent any potential leakage of GBDT-related information to users, we customize the random trees permutation protocol. This protocol fully capitalizes on GBDT's inherent properties to obscure the connection between interim results and trees, all while incurring minimal overhead. A thorough evaluation of these technical innovations is comprehensively presented. Our noteworthy contributions are succinctly outlined below:

- We propose a secure and effective framework for outsourcing GBDT inference, which protects the privacy of models, feature vectors, and predictions.
- Our approach involves a specialized algorithm for the proprietary GBDT. We also customize the secure comparison protocol for internal node determination using additively homomorphic encryption exclusively. Furthermore, we design the random trees permutations protocol using the properties of GBDT to effectively thwart any attempts at unauthorized information extraction. To solidify the integrity of our approach, we provide formal security proofs for these protocols.
- We conduct an empirical evaluation with GBDT of practical sizes, showing the actual performance of our security design, as well as substantial performance advantages for inference process compared with other existing privacy-preserving schemes (up to 36× in computation and 69× in communication).

II. Related Work

There have been massive works concerned with privacy-preserving decision tree evaluation (PPDE). Generally, there exist two kinds of technologies adopted in this area, i,e., (1) perturbation-based approaches and (2) cryptography-based approaches. Perturbation-based approaches [11] [12] [13] [14] typically involve introducing randomness or noise into the original database to safeguard the privacy of individual data entities. In general, these methods are efficient but with low accuracy.

In contrast, cryptography-based technologies can provide stronger privacy and accuracy. Our current focus primarily centers on this category of approaches. Ma et al. [28] proposed a privacy-preserving random tree framework with Paillier cryptosystem, which implemented accurate and secure training over encrypted data. Similarly, Wu et al. [29] devised an innovative solution for privacy-preserving vertical decision tree training and prediction, ensuring the non-disclosure of intermediary information. Furthermore, Liu et al. [30] proposed a federated extreme gradient boosting scheme that supports forced aggregation. The above schemes all adopt the HE-based mechanism, which is feasible for decision trees with privacy preservation. However, the secure computation implemented over a substantial volume of encrypted data results in a significant computational overhead.

To address the above issue, a model sharing-based privacy-preserving decision tree framework has been designed, which outsources encrypted model parameters rather than a large number of local data. Table I summarizes the strengths and weaknesses of the current work. Cock et al. [8] proposed a PPDE scheme in the semi-honest model. This scheme is designed by adopting Shamir's secret sharing scheme in the commodity-based model. Tai et al. [22] showed that their work successfully avoids an exponential number of encryptions in the depth of the tree. Tueno et al. [23] devised a method to represent the decision tree as an array and employed oblivious array indexing. In their scheme, garbled circuits (GC) [31], oblivious transfer (OT) [32], or oblivious RAM (ORAM) [33] are adopted. However, as shown in Table I the existing decision tree-based methods [8], [10], [21], [22], [23], [24], [25] face limitations in directly supporting GBDT. A straightforward approach involves transmitting ciphertexts containing predictions from all decision trees to the client, enabling the client to recover individual inference outcomes and subsequently aggregate them. Despite its simplicity, this approach raises concerns as it inadvertently exposes individual decision tree predictions to the client. The

adversary can discern sparse trees arriving "too early", allowing them to identify influential features in each tree by refining their query data [26]. Additionally, Luo et al. [34] proposed a feature inference attack, that successfully reconstructs private features by exploiting local linear correlations between the inputs and outputs of a single tree, even without prior background knowledge. The majority of existing single-server schemes [27], [22], [8], [16] necessitate traversing all nodes, incurring high overhead. Alternatively, schemes traversing only the correct paths typically require two or more servers [20], [19], posing deployment challenges in real-world scenarios.

To support GBDT, Bost et al. [7] built a privacy-preserving protocol for GBDT evaluation. They treated each decision tree as a polynomial and used fully homomorphic encryption. Kiss et al. [9] systematically categorized constant-round protocols to identify optimal instantiations, utilizing garbling techniques and homomorphic encryption. Zheng et al. [18] proposed a secure framework for outsourcing decision tree inference with huge communication overhead on the cloud side. However, the findings in Table I show that none of these works take into consideration the structure privacy of the model, which mainly consists of the architecture of the model and the real prediction paths.

In order to protect both node privacy and structure privacy, Wu et al. [27] enhanced Bost et al.'s approach by employing additively homomorphic encryption. Joye et al. [16] formulated protocols for privately evaluating decision trees through oblivious transfer. Nonetheless, both of these efforts still necessitate traversing each internal node ("One path" in Table I), incurring a notable overhead. Some subsequent work has investigated obtaining correct predictions through only one pathway. Ma et al. [19] proposed two MPC-based protocols, demanding two non-colluding servers for both complete and sparse trees. Bai et al. [20] designed a sublinear PPDE protocol involving $8d$ rounds of communication, which is larger than our $2d$ round. In addition, in Bai et al.'s work, the model owner needs to be engaged in the user's prediction process all the time. Note that these endeavors do not satisfy the one server requirement in Table I and require multiple non-colluding servers, which is very impractical and involves substantial overhead between servers.

III. Preliminaries

In this section, we introduce foundational concepts that form the basis of our approach. Table II provides a comprehensive list of the key notations used throughout this paper.

A. Gradient Boosting Decision Tree

GBDT, an ensemble machine learning algorithm, operates on a training dataset $D = \{(x_1, y_1), \cdots, (x_N, y_N)\}$ where $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and $y_i \in \mathcal{Y} \subseteq \mathbb{R}$. Using a specified loss function $\mathcal{L}(y, f(x))$, GBDT aims to discover an estimation function $\hat{f}(x)$ that maps every vector x_i to the corresponding label y_i to minimize the expected value of the loss function $\mathcal{L}$.

GBDT encompasses two kinds of nodes: non-leaf nodes (i.e., internal nodes) housing feature index and threshold information, and leaf nodes containing predicted values. During the prediction process, GBDT aggregates the predictions from all trees to generate the final prediction. We define the privacy aspects of GBDT as follows:

- **Node Privacy**: Node privacy consists of two kinds of node information in GBDT. Firstly, there are many internal nodes in decision trees, in which we want to protect the feature index and threshold information. Secondly, it extends to the leaf nodes, which store the predicted values of decision trees and determine the final GBDT prediction.
- **Structure (Path) Privacy**: The decision tree traverses the leaf nodes starting from the root node to generate predictions. This process defines a prediction path in the decision tree, referred to as path privacy, which must be safeguarded against disclosure. Furthermore, in the case of sparse trees, the structure of the tree should also be kept confidential.

TABLE II: Notations Used

Notations	Definition
n	Number of features
d	Maximum depth of decision trees
m	Number of decision trees in GBDT
γ	Maximum number of internal nodes in GBDT
f	Feature index
$e_{f,n}$	column vector in $\mathbb{R}^n$ with all zeros except for a 1 in position f
θ	Threshold value
$v_{i,j}$	The j-th node of the i-th tree
r_i	The random number of the i-th tree
pk/sk	Public/Secret key of AHE
$\boxplus$ / $\boxminus$	Addition / Subtraction of ciphertext
$[\![x]\!]$	Ciphertext of x
$F_{i,j}(x)$	Feature selection function of the $v_{i,j}$
$[\![x_{i,j}]\!]$	Encrypted data after feature selection of the $v_{i,j}$
CPT	Cipher Private Tree
SCP	Secure Comparison Protocol
RTP	Random Trees Permutation
Anti-RTP	Recovery algorithm for RTP

B. Pseudorandom Function

The standard definition for pseudorandom function (PRF) is given as follows:

Definition 1. Let $F : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be an efficient, length-preserving, keyed function. We say F is a *pseudorandom function* if for all probabilistic polynomial-time distinguishers *D*, there exists a negligible function *negl* such that:

$$|Pr[D^{F_k(\cdot)}(1^n) = 1] - Pr[D^{f_n(\cdot)}(1^n) = 1]| \leq negl(n),$$

where $k \leftarrow \{0,1\}^n$ is chosen uniformly at random and f_n is chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings.

C. Additively Homomorphic Encryption

Our scheme relies on additively homomorphic encryption (AHE), which supports fast addition and scalar multiplication. The standard definition for AHE is given as follows:

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2024.3395928

Definition 2. (**Additively Homomorphic Encryption [35]**): AHE schemes consist of the following (possibly probabilistic) poly-time algorithms.

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk)$: pk is the public key, while sk is the secret key.
- $\mathsf{Enc}(pk, m) \rightarrow [\![m]\!]$: probabilistic encryption algorithm produces $[\![m]\!]$, the ciphertext of message m.
- $\mathsf{Dec}(sk, [\![m]\!]) \rightarrow m$: decryption algorithm returns message m encrypted in $[\![m]\!]$.
- $\mathsf{Add}([\![m]\!], [\![m']\!])$: : The output is the encryption of plaintext addition/subtraction $(m \pm m')$.
- $\mathsf{DecA}(sk, [\![m \pm m']\!])$: decrypting $[\![m \pm m']\!]$ to obtain an addition/subtraction of plaintexts.

Definition 3. (**CPA security**): An AHE scheme is indistinguishability under chosen plaintext attack (IND-CPA) secure if, for any polynomial-time adversary $\mathcal{A}$, it holds that

$$\begin{aligned} \mathrm{Adv}_{\mathrm{CPA}}[\mathcal{A}] \triangleq |\Pr[\mathcal{A}(\mathrm{pk}, \mathsf{AHE.Enc}_{\mathrm{pk}}(0) = 1)] \\ - \Pr[\mathcal{A}(\mathrm{pk}, \mathsf{AHE.Enc}_{\mathrm{pk}}(1) = 1)]| = \mathrm{negl}(\lambda), \end{aligned} \tag{1}$$

where $(pk, sk) \leftarrow \mathsf{AHE.KeyGen}(1^\lambda)$.

D. Random Permutation

A random permutation f of order n is a random bijection over the set $\{1, 2, \ldots, n\}$, which can be denoted as

$$f = \begin{pmatrix} 1 & 2 & \ldots & n \\ f(1) & f(2) & \ldots & f(n) \end{pmatrix},$$

where $(f(1), f(2), \ldots, f(n))$ is a random reordering of $(1, 2, \ldots, n)$. The well-known Fisher-Yates shuffle [36] [37] iterates a sequence from the end to the beginning (or vice versa), and for each location i, it swaps the value at i with the value at a random target location j.

IV. Problem Statement

A. System Architecture

Our system architecture of outsourced GBDT inference is illustrated in Figure 1. Within this proposed framework, three distinct entities play pivotal roles: the Model Owner (MO), the Cloud Server (CS), and the $User$. A $User$ transmits encrypted personal data (e.g., age, weight, height, and traffic) to the CS for secure prediction because the CS is scalable and easy to manage the task (e.g., disease diagnosis or intrusion detection). On the CS side, a private GBDT algorithm is deployed to classify the $User$'s data. In order to safeguard $User$'s privacy, the CS remains oblivious to the actual results during the inference process, only returning ciphertext predictions. Furthermore, the GBDT model parameters should be prevented from being learned by the CS and the $User$ as this model is considered proprietary to the MO.

B. Threat Model

Similar to most of the existing privacy-preserving machine learning efforts [38] [39] [20] [17], we focus on the *honest-but-curious* adversary who will not deviate from the defined protocols but will try to learn all possible information from legitimately received messages. We assume each party will not collude with others. In our threat model, the CS will try to recover the encrypted feature vectors as well as the encrypted GBDT classifier. The $User$ will try to infer the structure of the tree in GBDT from the information returned by the CS. Like all previous work [27] [19], we do not protect the following generic parameters about the GBDT: the max depth d of GBDT, the number γ of internal nodes, and the dimension n of feature vectors. The following privacy requirements should be guaranteed.

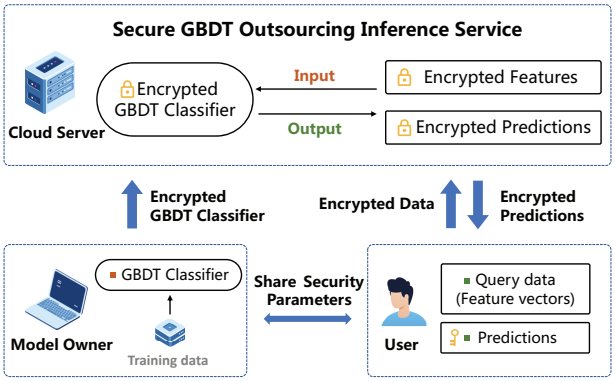

Fig. 1: System architecture

- **Data privacy**: The feature vectors and prediction results should be protected from the CS. Namely, the $User$ needs to upload encrypted private data for secure online inference and then get the corresponding encrypted prediction results. All ciphertexts should not be recovered by the CS.
- **Classifier privacy**: Specifically, classifier privacy includes both node privacy and structure privacy, as we stated in Preliminaries. Therefore, the GBDT classifier should be kept secret from the CS and the $User$. In other words, the CS cannot reveal the corresponding plaintext form of the encrypted model. And the $User$ cannot infer information about the GBDT based on the prediction results.

V. Efficient and Privacy-preserving Outsourcing of GBDT Inference

A. High-level Overview

We propose an efficient and privacy-preserving framework for outsourcing GBDT inference. From a high-level perspective, our scheme comprises three essential phases:

(1) **Private Tree Transformation**: This phase focuses on safeguarding the MO's private model by effectively concealing the node and structure information of GBDT.
(2) **Secure Comparison Protocol**: This step involves determining internal node results between the CS and the $User$, with a potential risk of prediction path leakage.
(3) **Random Trees Permutation Protocol**: To mitigate this risk and enhance model evaluation efficiency, we design the random tree permutation protocol.

The overall scheme can be divided into two parts. First, the GBDT must be transformed before the MO uploads the model (steps 1 and 2 in Figure 3). Then, by deploying the encoded GBDT on the cloud, the two protocols collaborate to achieve GBDT prediction (step 3-6 in Figure 3). Next, we will delve into each phase with detailed elaboration.

B. Private Tree Transformation

Algorithm 1 Private Tree Transformation

Input: The root node v of each tree in GBDT, AHE.pk, PRF, a seed s_{MO} of PRF
Output: The encoded private $GBDT$

1: **Note:** T_i denotes the i-th tree in GBDT and $T_i.d$ is the maximum depth of the tree. A node v is a data structure including the following fields: $v.value$ is predicted result when v is a leaf node; $v.index$ denotes the index of node v; $v.sf$ denotes the switch flag of node v; $v.d$ denotes the depth of node v. $rand\_f$ and $rand\_\theta$ are two random values for feature index and threshold. In addition, $v.left$ and $v.right$ are denoted as the left and right sub-trees of v, respectively.
2: **function** tree_transformation(v, d)
3: **if** v is a leaf **and** $v.d = d$ **then**
4: encrypt its value: AHE.Enc($pk, v.value$);
5: **end if**
6: **if** v is a non-leaf node **then**
7: encrypt threshold value: AHE.Enc($pk, v.\theta$);
8: mask feature index: $e'_{v.f,n} = e_{v.f,n} + F(s_{MO}, e_{v.index,\gamma})$;
9: tree_transformation($v.left, d$);
10: tree_transformation($v.right, d$);
11: **end if**
12: **if** v is a leaf **and** $v.d \neq d$ **then**
13: randomly select 0 or 1 for $split$;
14: **if** $split == 1$ **then**
15: select $rand\_f$ and $rand\_\theta$;
16: create a dummy node v' with $F(s_{MO}, e_{v'.rand\_f,\gamma})$ and AHE.Enc($pk, v'.rand\_\theta$);
17: set left child and right child of v' to node v;
18: attach the created node to the parent node;
19: **end if**
20: **end if**
21: **end function**
22: **for** $i \leftarrow 1$ to m **do**
23: randomly disrupt the index of non-leaf nodes in Tree T_i;
24: tree_transformation($T_i.root, T_i.d$);
25: **for** $j \leftarrow 1$ to γ **do**
26: randomly select 1 or -1 for switch flag $v_{i,j}.sf$;
27: **if** $v_{i,j}.sf == -1$ **then**
28: swap left and right child nodes;
29: **end if**
30: encrypt switch flag: AHE.Enc(pk, $v_{i,j}$.sf);
31: **end for**
32: **end for**

There are several crucial aspects when dealing with GBDT under privacy settings, such as information about the nodes of each tree in GBDT (e.g., thresholds, feature indexes, leaf values, etc.). Moreover, even if the nodes and feature vectors are well protected (e.g., by encryption), revealing tree structure and prediction paths can leak sensitive information [9] [26]. To avoid leaking the above GBDT privacy, the private tree transformation algorithm π_{PTT} is proposed. Notice that unlike prior works [9] [18], we comprehensively protect the **node** and **structure** privacy of GBDT. We modify GBDT as shown in Algorithm 1.

For **node privacy**, we use AHE to encrypt the prediction value of leaf nodes that arrive at the max depth d and the threshold value of internal nodes (**line 3-7**). Moreover, to ensure secure feature selection, we introduce the pseudorandom function (PRF) to mask the feature index of internal nodes (**line 8**). Specifically, the feature index f is an index in {1, 2, ..., n}, and each internal node $v_{i,j}$ identifies the data $x_{i,j}$ used for the boolean test function with the function $F_{i,j}(x) = e^T_{v_{i,j}.f,n} \cdot x$, where $\cdot$ is the standard row-by-column multiplication. Here the vector $e_{f,n}$ is the column vector in $\mathbb{R}^n$ with all zeros except for a 1 in position f and $e^T_{f,n}$ is its transpose. In order to hide the information of f and to keep the function $F_{i,j}(x)$ available, we design a secure feature selection method for each internal node. Let γ be the number of internal nodes, we disrupt the index $\{1, 2, ..., \gamma\}$ and assign the disrupted index to each internal node in level order traversal. Now, we assume that $F : \{0,1\}^\gamma \times \{0,1\}^\gamma \rightarrow \{0,1\}^n$ is a public PRF. The MO sample a seed $s_{MO} \leftarrow \{0,1\}^\gamma$. And for internal node $j = 1, 2, ..., \gamma$, the MP computes $e'_{v_{i,j}.f,n} = e_{v_{i,j}.f,n} + F(s_{MO}, e_{v_{i,j}.index,\gamma})$ in each tree ($i = 1, 2, ...m$). Therefore, the MO hides the feature index f by adding the output of PRF on the vector $e_{f,n}$. After these operations, the feature index of each internal node f becomes vector $f' = e'_{f,n}$. As a result, the function $F_i(x) = e'^T_{f,n} \cdot x$. Without s_{MO}, the adversary cannot derive information about f based on f'.

For **structure privacy**, we randomly replace the leaf node v (only once) that does not reach the maximum depth with a dummy node (random feature index and threshold) and redirect the left and right children of the random node to v (**line 12-20**). After dealing with node v, we recursively process its left and right children to form the function tree_transformation. We randomly disrupt the index of internal nodes in T_i by using uniform permutation $\pi_i()$. In addition, during level order traversal, the MO randomly assigns switch flag 1 or -1 to each internal node. The left and right child nodes are exchanged if the switch flag sf is -1. After this, the sf is encrypted by AHE (**line 25-31**).

Unlike the previous approach [7] [9], our algorithm π_{PTT} does not pad each GBDT tree to be complete or near-complete and run comparisons for all internal nodes. This is because we have designed a customized protocol in Section V-D to prevent the $User$ from locating trees that reach leaf nodes "too early". Note that we encode the structure of GBDT by expanding only a part of the leaf nodes and randomly flipping branches, so it is efficient in the inference process, especially when evaluating large trees.

Security. Here we briefly describe the security of preprocessing for GBDT. It is easy to infer that the above transformation for GBDT is secure against the semi-honest model owner MO and cloud server CS.

Specifically, for the model owner MO, since the entire preprocessing process does not require the participation of the cloud server and user, the MO cannot obtain any private information about the CS and $User$. Similarly, for the cloud server CS, since the transformation is non-interactive and the encoded GBDT satisfies the IND-CPA security defined in Section III-C, CS cannot obtain the plaintext corresponding to the ciphertext sent by the model owner. In addition, the feature index in each internal node is masked by a PRF. Since the PRF inherits the efficiently checkable property of a random function (see *Pseudorandom Function* subsection), it is difficult for CS to obtain the feature index without s_{MO}. Therefore, CS cannot obtain any useful information from nodes, otherwise, he can distinguish the encrypted GBDT from random.

The structure privacy refers to the fact that the original structure of the model cannot be deduced. For any probabilistic polynomial adversary, the probability of guessing the tree structure for any tree of depth d is computationally indistinguishable from the probability of randomly guessing from the set of all

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2024.3395928

possible tree structures of depth d. We protect the structure privacy in two key steps. Firstly, we implement random splits for leaf nodes that have yet to reach the maximum depth. Secondly, we employ a randomized swapping of branches for each internal node, utilizing random switch flags to interchange left and right branches.

To demonstrating the effectiveness of our algorithm for structural protection, we theoretically calculate the probability that the adversary guesses the original tree structure in three different situations. Firstly, in the general case, the number of internal and leaf nodes within the left and right branches exhibits similarity, while the specific structure may vary. We assume only a quarter of the leaf nodes in both the left and right branches achieve layer $(d+1)$, and the number of non-leaf nodes in layer d is half of the leaf nodes in layer $(d+1)$. When the practical depth $d = 9$, the probability of the adversary accurately guessing the architecture is less than or equal to 2^{-97}, which is considered negligible. Secondly, in the unbalanced case, the left and right branches exhibit an imbalance, with the assumption that the left branch possesses a significantly larger number of nodes than the right branch. We also assume only a quarter of the leaves of the left branch reach layer $10(d = 9)$, and the right branch has no nodes reaching maximum depth. The probability of the adversary correctly guessing the architecture is less than 2^{-49}, also deemed negligible. Finally, we consider a perfect binary tree with depth $d = 9$. Only the random splits at layer d need to be considered in this case, and the random branch swapping is invalidated. The total number of combinations for all possible architectures is given by $c = 2^{2^{d-1}}$. Consequently, the probability that the adversary can correctly guess the architecture is 2^{-256}, which is negligible. In summary, our private tree transformation algorithm ensures the confidentiality of both individual **node** and the original **structure**.

C. Secure Comparison Protocol

In this section, we define integers in the interval $[-2^{\ell-1}, 2^{\ell-1})$. Our protocol operates over a modulo field $\mathbb{Z}_n$, which $n \geq 2^{\ell}$ and is either 2^k or a prime p. For any integer a belonging to the range $[-2^{\ell-1}, 2^{\ell-1})$, we can map it to the corresponding element in the ring of integers modulo n (denoted as $\mathbb{Z}_n$) by calculating $a \bmod n \in \mathbb{Z}_n$.

A naive way of comparing two integers a and b would be $c = (a - b)$ and see whether c is positive. However, knowing the difference c and one of the values from a or b will reveal the other value. Hence, besides encrypting a and b using AHE, the difference c also needs to be blinded through multiplication with a much larger random number r. To efficiently hide a number of sizes $O(m)$ by multiplication, the size of random hiding factor r for multiplication has to be at least $O(m^2)$.

However, if the difference between a and b is 0, i.e., they are equal, the result of $(a - b) * r$ will be 0 regardless of the chosen hiding factor r. This vulnerability can be avoided by subtracting a smaller random value $r' < r$ that does not change the result. Note that both the r and r' are positive and r' is strictly smaller than the hiding factor r.

If the number a and b are taken from the domain $\mathcal{D} = [l, h]$, then the difference $(a-b)$ is in the domain $\mathcal{D}_{sub} = [l-h, h-l]$. Let l_r be $(l-r)$ and h_r be $(h-r)$. We can choose the random numbers r from the domain $\mathcal{D}_r = [1, (h-r)^2] = [1, h_r^2]$. The another random numbers r' come from the domain $\mathcal{D}_{r'} = [0, r]$. To run the protocol correctly, we have to prevent "overflow" modulo n. In the worst case, the positive value can be close to $h_r * h_r^2 + h_r^2$. Thus the modulus n of the additively homomorphic encryption scheme needs to be larger than $2 * (h_r^3 + h_r^2)$ because we set the upper half of the range $[0, n-1]$ to be negative numbers.

There is a small leak in the protocol that occurs with very minor probability. One of the parties knows the information of the comparison if c=0, (i.e., a=b). Let E_0 denote the event in which the comparison result c is equal to 0. We now calculate the probability $Pr[E_0]$ that event E_0 happens. If $c = 0$, if and only if $a = b$ and for every r, there is $r' = 0$. We set the probability of $a = b$ as p, and p is definitely less than 1. If r and r' are chosen uniformly from $\mathcal{D}_r$ and $\mathcal{D}_{r'}$, respectively, then probability $Pr[E_0]$ of revealing $a = b$ is:

$$\begin{aligned} Pr[E_0] &= p * (\frac{1}{h_r^2} * \frac{1}{1} + \frac{1}{h_r^2} * \frac{1}{2} + \cdots + \frac{1}{h_r^2} * \frac{1}{h_r^2}) \\ &= p * \sum_{i=1}^{h_r^2} \frac{1}{h_r^2} * \frac{1}{i} \approx p * \frac{2 \ln h_r}{h_r^2} < \frac{2 \ln h_r}{h_r^2} \end{aligned} \tag{2}$$

For large numbers h_r, the probability $Pr[E_0]$ is negligible. E.g., when comparing 32-bit numbers, $p < 2^{-58}$. This can be made even more difficult by choosing a distribution at random to pick r and r'.

In our outsourcing scenario, the comparative information is provided by the $User$ and the MO. To decide which path to go between two children in an internal node $v_{i,j}$, the CS uses the encrypted threshold value $[\![v_{i,j}.\theta]\!]$ and masked feature index $v_{i,j}.f'$ to compare with encrypted testing data $[\![x_{i,j}]\!]$. The protocol π_{SCP} is executed by the CS and $User$ and is formulated as given in Figure 2. We elaborate on the protocol π_{SCP} as follows.

Data Encryption Phase. To avoid query sample x leakage, the $User$ needs to prepare encrypted data $[\![x]\!] = \mathsf{AHE.Enc}(pk, x)$. In addition, the $User$ calculates $x_j = -F(s_{MO}, e_{j,\gamma})^T \cdot x$, which $j = 1, 2, ..., \gamma$. For each vector x_j, $User$ sums the elements in the vector and then encrypts them, i.e. $[\![x'_j]\!] = \mathsf{AHE.Enc}(pk, \sum_{i=0}^{n-1} x_j[i])$. The goal of γ ciphertext is to complete secure feature selection. Then, the $User$ sends the encrypted data $[\![x]\!]$, $[\![x'_1]\!]$, ..., $[\![x'_\gamma]\!]$ to the server.

Secure Feature Selection Phase. Since the MO sends the CPTs, for each non-leaf node $v_{i,j}$, the CS has the encrypted threshold value $[\![v_{i,j}.\theta]\!]$ and masked features $v_{i,j}.f'$. The CS computes $[\![x_{i,j}]\!] = F_i([\![x]\!]) + [\![x'_j]\!] = {e'}^T_{v_{i,j}.f,n} \cdot [\![x]\!] + [\![x'_j]\!] = (e^T_{v_{i,j}.f,n} + F(s_{MO}, e^T_{j,\gamma})) \cdot [\![x]\!] + [\![-F(s_{MO}, e^T_{j,\gamma}) \cdot x]\!] = e^T_{v_{i,j}.f,n} \cdot [\![x]\!]$, which offset the noise of the PRF, enables secure feature selection, and the resultsing data is ciphertext. The secure comparison protocol for two ciphertext ($[\![x_{i,j}]\!]$ and $[\![v_{i,j}.\theta]\!]$) can now be executed.

Masked Comparison Phase. As we mentioned before, the server calculates $[\![c_i]\!] = ([\![x_{i,j}]\!] - [\![v_{i,j}.\theta]\!]) \cdot r_i + r'_i$ and form m ciphertext comparison results as $[\![C]\!]$. Next, we generate random flipping s_i for each result to secure the real prediction results. The CS sends the ciphertext $[\![C']\!]$ to the $User$. After

Preamble: Consider a Gradient Boosting Decision Tree (GBDT) consisting of m decision trees. After the private tree transformation, there are m cipher private trees (CPTs). Determine the children of m nodes in these CPTs simultaneously.
Input: The server holds m nodes from CPTs, each node $v_{i,j}$ includes masked feature index $v_{i,j}.f'$ and encrypted threshold value $[\![v_{i,j}.\theta]\!]$. The user holds query sample x, a seed s_{MO} of PRF.
Output: The server obtains Res containing m comparison results.
Protocol:

1. Data Encryption Phase:
   - The user computes $[\![x]\!] \leftarrow \mathsf{AHE.Enc}(pk, x)$ to encrypt the query sample.
   - The uesr computes γ masked query samples: $\{x_j\}_{j=1,\dots,\gamma} \leftarrow \{-F(s_{MO}, e_{j,\gamma})^T \cdot x\}_{j=1,\dots,\gamma}$.
   - The user encrypts the sum of n elements for each masked query sample: $\{[\![x'_j]\!]\}_{j=1,\dots,\gamma} \leftarrow \{\mathsf{AHE.Enc}(pk, \sum_{i=0}^{n-1} x_j[i])\}_{j=1,\dots,\gamma}$.
   - The user sends the encrypted samples $[\![x]\!]$, $[\![x'_1]\!]$, ..., $[\![x'_\gamma]\!]$ to the server.
2. Secure Feature Selection Phase:
   - For each node $v_{i,j}$, the server performs secure feature selction to obtain encrypted data $[\![x_{i,j}]\!]$ corresponding to the feature $v_{i,j}.f$:

$$[\![x_{i,j}]\!] = F_i([\![x]\!]) + [\![x'_j]\!] = e'^T_{v_{i,j}.f,n} \cdot [\![x]\!] + [\![x'_j]\!] = (e^T_{v_{i,j}.f,n} + F(s_{MO}, e^T_{j,\gamma})) \cdot [\![x]\!] + [\![-F(s_{MO}, e^T_{j,\gamma}) \cdot x]\!] = e^T_{v_{i,j}.f,n} \cdot [\![x]\!]$$

3. Masked Comparison Phase:
   - The server chooses random number r_i and r'_i with $0 \le r'_i < r_i$, and calculates the comparison results $[\![C]\!] = ([\![c_1]\!], \cdots, [\![c_m]\!])$, each of which is computed as follows:

$$[\![c_i]\!] = ([\![x_{i,j}]\!] - [\![v_{i,j}.\theta]\!]) \cdot r_i + r'_i$$

   - The server generates $S = (s_1, \cdots, s_m)$, where $s_i \in \{-1, 1\}$, to mask the results C. Get the masked results $[\![C']\!] = (s_1 \cdot [\![c_1]\!], \cdots, s_m \cdot [\![c_m]\!])$ and send the $[\![C']\!]$ to the user.
   - The user decrypts $[\![C']\!]$ and decides plaintext result $p_i = 1$ if and only if $c'_i \mod n < \frac{n}{2}$ (n is the ring of integers modulo). Then the user sends the bit results $P = (p_1, \cdots, p_m)$ for m nodes to the server. The following derivation shows this equivalence:

$$c'_i \mod n < \frac{n}{2} \Leftrightarrow c_i \cdot s_i > 0$$

   - The user assumes $s_i = 1$, then:

$$(x_{i,j} - v_{i,j}.\theta) \cdot r_i + r'_i > 0 \Leftrightarrow x_{i,j} - v_{i,j}.\theta > 0 > -\frac{r'_i}{r_i} > -1 \Leftrightarrow x_{i,j} > v_{i,j}.\theta \Rightarrow p_i = res_i \cdot s_i = 1$$

     The res_i represents the true comparison result, which is masked by s_i. Else $p_i = -1$ if and only if $c'_i \mod n \ge \frac{n}{2}$.
   - The user sends the $P = (p_1, \cdots, p_m)$ to the server.
4. Output Phase:
   - The server gets the real comparison results $Res = S \cdot P = (s_1 \cdot s_1 \cdot res_1, \cdots, s_m \cdot s_m \cdot res_m) = (res_1, \cdots, res_m)$ because of $s_i \cdot s_i = 1$.

Fig. 2: Our protocol π_{SCP} for secure comparison in GBDT

decrypting $[\![C']\!]$, the $User$ infer comparison result $p_i = 1$ if and only if $c'_i \mod n < \frac{n}{2}$. Importantly, the result p_i is concealed by the s_i, and the random flipping S also significantly reduces the probability of the user deriving the true predicted path. Therefore, the server deduces the actual comparison results, denoted as res_i, by multiplying p_i with s_i. If res_i equals 1, the current node in tree T_i should choose the left child for the next comparison; otherwise, it should choose the right child.

Output Phase. Finally, the CS obtains the true comparison result res_i by multiplying the received p_i by s_i again. Our proposed comparison protocol Π_{SCP} securely gets the comparison results for m non-leaf nodes in each round of communication. In our protocol, the probability of a small leak in a real-world scenario is negligible. The CS can only know the results because all the messages involved in the comparison are ciphertexts. On the other hand, the $User$ can not determine the real results because of masking. The difference between testing data and threshold is also fuzzed. Thus the User is only able to help calculate the comparison results without deducing other information. We assume that the CS's flipped random numbers are chosen uniformly, so the probability of a $User$ guessing the real path is $\frac{1}{2^d}$. This probability is not negligible, so we need to design another protocol to protect path privacy.

Security. Our protocol for securely generating comparison results, π_{SCP}, is secure against the semi-honest cloud server CS and user $User$. We provide the following theorem.

Theorem 1. *Let the additively homomorphic encryption and pseudorandom function used in π_{SCP} have the properties defined in Section III. π_{SCP} is secure against the semi-honest cloud server CS and user $User$.*

D. Random Trees Permutation Protocol

With the secure comparison protocol π_{SCP} at internal nodes, we now describe how to securely generate the prediction result without compromising path privacy. When the sequence of transmitted ciphertexts aligns with the trees, the user can obtain both comparison and prediction outcomes for each individual tree. In addition, the $User$ can locate the sparse trees that arrive at leaf nodes "too early" based on the length of the path. Recognizing that in GBDT each tree operates with a degree of independence during the prediction phase, we craft the random trees permutation protocol π_{RTPP} to obfuscate the relationship between trees and comparison results.

We use the algorithm in Section III-D to achieve randomness. The process of the random trees permutation (RTP) algorithm is as follows. First, the CS needs to generate a random bijection table R. Then, the comparison results are permuted according to the table. For example, for every $i \in [n]$, we put the s_i in the $f(i)$-th slot of a new vector $\mathbf{v}$, where $f(i) \in R$ (random

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2024.3395928

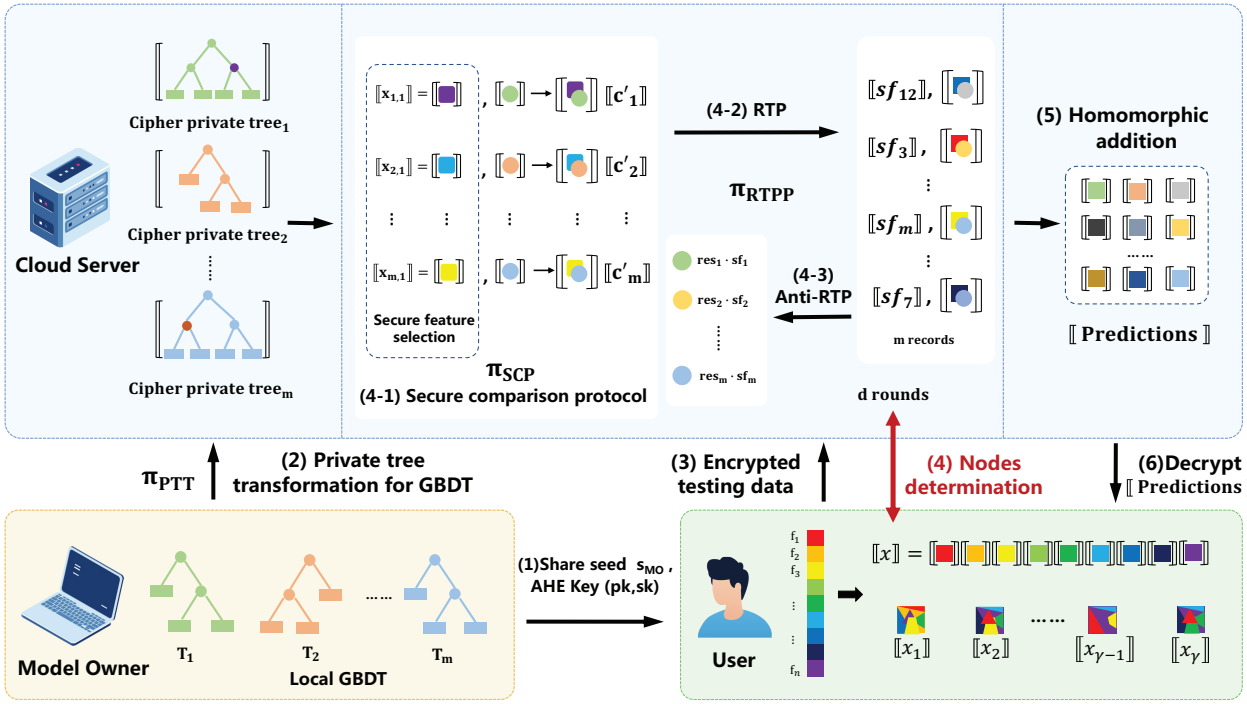

Fig. 3: System overview

bijection table). The recovery algorithm called Anti-RTP also relies on R. For instance, for every $i \in [n]$, we put the number $v_{f(i)}$ in the i-th slot.

Now, we introduce the process of our random trees permutation protocol π_{RTPP}. In each round, the CS uses the RTP algorithm to permute the secure comparison results. Then, the $User$ decrypts the disrupted ciphertext and sends it to the CS. By executing the Anti-RTP algorithm, the CS can restore the correct order to get the real result of each tree. Therefore, the CS knows which child node each tree should go to in the next round, and the $User$ does not know the child node. Notably, the semi-honest adversary cannot deduce the threshold value of each node even through adaptive queries. This is because we prevent the adversary from associating a specific tree with a particular result received in each round. Note that the branches of each tree in GBDT have been randomly swapped by switch flags, so the path obtained by CS is not the true predicted path. The protocol protects the node information of GBDT and also protects the path privacy. As we mentioned in Section V-C, with our random trees permutation protocol π_{RTPP} and considering sparse trees, the probability of a $User$ guessing the correct predicted path of a tree is reduced from $\frac{1}{2^{(d-1)}}$ to $\frac{1}{m(2^{(d+1)}-2)}$. And the probability that the user guesses the correct predicted path for all trees in GBDT is $\frac{1}{m!(2^{(d+1)}-2)}$ e.g., for practical setting $d = 8$ and $m = 32$, the probability is $\frac{1}{32!*510}$, which is negligible. Besides, the random permutation algorithm requires very few steps, each iteration requires only a random integer and a swap operation.

Security. Our protocol for obfuscating decision trees in GBDT, π_{RTPP}, is secure against the semi-honest cloud server CS and user $User$. We provide the following theorem.

Theorem 2. *Let the Fisher-Yates shuffle used in π_{RTPP} have the properties defined in Section III. π_{RTPP} is secure against the semi-honest cloud server CS and user $User$.*

E. Putting It All Together

Putting it all together, we summarize the workflow of our scheme in Figure 3.

Initialization The initialization phase includes three steps. In step 1 of Figure 3, the $User$ and the MO share PRF and a seed s_{MO}, pk, and sk. The MO trains local GBDT $(G_1, G_2, \cdots, G_n)$, encrypts them using the private tree transformation algorithm, and uploads all the CPTs to the CS in step 2. The $User$ encrypts his/her testing data in step 3. In particular, the encrypted testing data $[\![x]\!]$ and $\{[\![x'_1]\!]$, ..., $[\![x'_\gamma]\!]\}$.

Inference The inference phase includes three steps. Specifically, in step 4 of Figure 3, the CS computes the decision path for all trees by communicating with the $User$ (d rounds of nodes determination). The π_{SCP} and π_{RTPP} protocols are mainly executed in each round of interaction. The CS first executes secure feature selection and computes the masked comparison result c'_i for each tree under ciphertext. As shown in Figure 3, $[\![c'_1]\!]$ is computed from $[\![x_{1,1}]\!]$ and the encrypted threshold in CPT_1. If there exists a sparse tree t that arrives early, the CS can continue to perform the mask comparison phase in the protocol π_{SCP} by using the ciphertext $[\![x_{t,j}]\!]$ obtained in the last round. The c'_i and the corresponding switch flag sf_i together form the comparing values (c_i, sf_i). Then, the CS executes the RTP algorithm and generates a bijection table. By using the table, the CS obfuscates the corresponding relationships between the tree and encrypted comparing value. The CS sends the disrupted results to the user, note that the number of these results is equal to m as shown in Figure 3. Once the $User$ receives and decrypts all the results, he/she multiplies SCP results and corresponding switch flags, i.e., $(res_i \cdot s_i) \cdot sf_i$. The CS recovers the order

according to the bijection table (Anti-RTP) and array S for masking, i.e., $res_i \cdot sf_i$ in Figure 3. If a comparison result is 1, then the left child node should be selected, otherwise the right child node (-1). Then the CS chooses the next nodes to go for all trees. Because the structure of each tree is disordered by the switch flags, the result $(res_i \cdot sf_i)$ is equivalent to multiplying sf_i again in each encoded tree, i.e., $res_i \cdot sf_i \cdot sf_i = res_i$. Therefore, there is no need to alter the threshold values, and the correct predictions will be obtained.

In step 5 of Figure 3, the CS does a homomorphic addition to all the encrypted leaf values and sends the result to the $User$. We should note that some trees may reach leaf nodes earlier. So the CS can sum the results that reach the leaf nodes in advance. Rather than leaving the values of the leaf nodes to be calculated uniformly at the end, the inference time can be reduced. At last, in step 6 of Figure 3, the $User$ decrypts the ciphertext and obtains the real predictions.

Security. Our protocol for privacy-preserving outsourcing of GBDT inference is secure against the semi-honest model owner MO, cloud server CS, and user $User$. We provide the following theorem.

Theorem 3. *Let the additively homomorphic encryption, pseudorandom function, and Fisher-Yates shuffle used in our protocol have the properties defined in Section III. Our proposed privacy-preserving outsourcing of GBDT inference is secure against the semi-honest model owner MO, cloud server CS, and user $User$.*

F. Security Analysis

Before formally stating our proof, we introduce some notation. We will consider executions of our protocols where the underlying cryptographic primitive is instantiated with security parameter λ. The $\mathcal{MO}$ encodes GBDT which consists of a set T of m trees. There are at most γ internal nodes in each tree. In addition, the switch flag of an internal node is set to sf. The random permutation for node indexes is set to π. α is a seed to perform the AHE encryption. The seed of PRF is set to $s_{\mathcal{MO}}$. Denote the input of the user with x, and with x_i is computed by PRF.

Proof of Theorem 1 We first define the functionality of the protocol π_{SCP}, denoted as $\mathcal{F}_{SCP}$, as shown in Figure 4. Now we prove security for the semi-honest cloud server and then demonstrate security against semi-honest users.

Input: The $\mathcal{CS}$ holds m nodes from CPTs, each node $v_{i,j}$ includes masked feature index $v_{i,j}.f' \in \{0,1\}^n$ and encrypted threshold value $[\![v_{i,j}.\theta]\!]$. The $\mathcal{U}ser$ holds query sample x, each element in $x \in \mathbb{Z}_N$, a seed $s_{\mathcal{MO}} \in \{0,1\}^\gamma$ of PRF.
Output: The $\mathcal{CS}$ obtains $Res = \{res_i = 1\ (x_{i,j} > v_{i,j}.\theta)$,otherwise $-1\}_{i=1,...,m}$.

Fig. 4: Functionality of the secure comparison $\mathcal{F}_{SCP}$

Semi-honest cloud server security. The simulator $\mathrm{Sim}_{\mathcal{CS}}$ samples random number r^* from $\mathbb{Z}_{N^2}$, and the x is drawn from $\mathbb{Z}_N$. The another random numbers r'^* come from the domain $[0, r^*]$. The simulator and the user run the secure comparison protocol to generate the comparison results for internal nodes. When the simulator accesses the ideal functionality, it provides Res as output. We now show the indistinguishability between real and simulated views by the following hybrid arguments.

- $\mathtt{Hyb}_1$ This data encryption and secure feature selection phase are identical to `Real`.
- $\mathtt{Hyb}_2$ Please note that in this hybrid the simulator $\mathrm{Sim}_{\mathcal{CS}}$ can access $\mathcal{U}ser$'s input $[\![x_{i,j}]\!]$ corresponding to the feature index $v_{i,j}.f$. Therefore, for $i = 1, ..., m$, the simulator $\mathrm{Sim}_{\mathcal{CS}}$ sample a random value r_i^* chosen from $\mathbb{Z}_{N^2}$ uniformly. Another random value $r_i'^*$ is chosen from the domain $[0, r_i^*]$ uniformaly. $\mathrm{Sim}_{\mathcal{CS}}$ computes $([\![x_{i,j}]\!] - [\![v_{i,j}.\theta]\!]) \cdot r_i^* + r_i'^*$ and obtains the $[\![c_i]\!]^*$ for $i = 1, ..., m$. Moreover, the simulator $\mathrm{Sim}_{\mathcal{CS}}$ generate $s_i^* \in \{1, -1\}$. After masking $[\![c_i]\!]^*$ with s_i^*, $\mathrm{Sim}_{\mathcal{CS}}$ sends the result $[\![C']\!]^*$ to the $\mathcal{U}ser$. The $\mathcal{U}ser$ follows the protocol and sends the result P^* to $\mathrm{Sim}_{\mathcal{CS}}$.
- $\mathtt{Hyb}_3$ Instead of using the original $S = (s_1, ..., s_m)$ to get the real comparison results, the $\mathrm{Sim}_{\mathcal{CS}}$ multiply each element in P^* with s_i^*. Since the $s_i^* \cdot s_i^* = 1$, the output of $\mathrm{Sim}_{\mathcal{CS}}^{\pi_{\mathrm{SCP}}}([\![\mathrm{x}]\!], [\![\mathrm{x}_1]\!], ..., [\![\mathrm{x}_\gamma]\!])$ is identically distributed to the view $\mathrm{Real}_{\mathcal{CS}}^{\pi_{\mathrm{SCP}}}$ of the corrupted cloud server.

Semi-honest user security. The simulator $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ replaces the original $seed$ by a random number $seed^*$ to perform the AHE. In addition, $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ generates a new seed $s_{\mathcal{MO}}^*$ for the PRF. Note that we assume the $seed^*$ and $s_{\mathcal{MO}}^*$ is also used by the model owner in the private tree transformation. Then, $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ uses the randomness to generate the encrypted samples to complete the secure comparison protocol with the cloud server.

- $\mathtt{Hyb}_1$ In this hybrid, $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ encrypted query sample with uniformly random number $seed^*$ and a $s_{\mathcal{MO}}^* \in \{0,1\}^\gamma$. The simulator $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ sends the ciphertext $[\![x]\!]^*, [\![x_1]\!]^*, ..., [\![x_\gamma]\!]^*$ to the cloud server. The distributions of ciphertext with the same message and pk but different seeds are indistinguishable with semantic security. According to our assumption, the feature index and threshold of node $v_{i,j}$ in CPT_i have become $[\![v_{i,j}.f]\!]^*$ and $[\![v_{i,j}.\theta]\!]^*$. The IND-CPA secure AHE and PRF ensures this hybrid is indistinguishable from real protocol.
- $\mathtt{Hyb}_2$ This secure feature selection phase is identical to `Real`.
- $\mathtt{Hyb}_3$ The $\mathcal{CS}$ follows the protocol and computes $[\![c_i]\!]^* = ([\![x_{i,j}]\!]^* - [\![v_{i,j}.\theta]\!]^*) \cdot r_i + r_i'$. After masking $[\![c_i]\!]^*$ with $s_i \in \{1, -1\}$, $\mathcal{CS}$ send the result $[\![C']\!]^*$ to the $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$. The simulator $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ decrypts $[\![C']\!]^*$ to get the result p_i^*. Because we only change the AHE and PRF seeds, the additivity of AHE ensures the $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ return the same sign bit of $c_i'^*$. Therefore, the output of $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}^{\pi_{\mathrm{SCP}}}$ is identically distributed to the view $\mathrm{Real}_{\mathcal{U}\mathrm{ser}}^{\pi_{\mathrm{SCP}}}$ of the corrupted user.

The proof of Theorem 1 is completed.

Proof of Theorem 2 Let $\mathcal{F}_{RTPP}$ shown in Figure 5 be the functionality of obfuscating decision trees in GBDT.

Input: The $\mathcal{CS}$ holds ciphertext $[\![C']\!] = \{[\![c_1']\!], ..., [\![c_m']\!]\}$ from π_{SCP}.
Output: The $\mathcal{CS}$ obtains comparison results $Res = \{res_1, ..., res_m\}$.

Fig. 5: Functionality of the random trees permutation $\mathcal{F}_{RTPP}$

Semi-honest user security. The security of the protocol π_{RTPP} against the semi-honest user $\mathcal{U}ser$ is evident by observing the execution of the protocol. This stems from the fact that $\mathcal{U}ser$ receives randomly disturbed ciphertext. And every comparison result is masked by a random value $s_i \in \{1, -1\}$. $\mathcal{U}ser$ has no way of knowing how the results correspond to each tree. This also implies that for trees that reach leaf nodes "too early", $\mathcal{U}ser$ is unable to derive which are sparse trees in GBDT. Naturally, the predicted paths for each tree cannot be derived. With our random trees permutation protocol π_{RTPP} and considering sparse trees, the probability of a $User$ guessing the correct predicted path of a tree is reduced from $\frac{1}{2^d}$ to $\frac{1}{m(2^{(d+1)}-2)}$. And the probability that the user guesses the correct predicted path for all trees in GBDT is $\frac{1}{m!(2^{(d+1)}-2)}$ e.g., for practical setting $d = 8$ and $m = 32$, the probability is $\frac{1}{32!*510}$, which is negligible. Next, we focus on the security analysis of π_{RTPP} against semi-honest cloud server $\mathcal{CS}$.

Semi-honest cloud server security. The simulator $\mathrm{Sim}_{\mathcal{CS}}$ replaces the original permutation f with f^*. Then, $\mathrm{Sim}_{\mathcal{CS}}$ uses the bijection table to obfuscate comparison results and get the results for node determination in the correct order.

$\mathtt{Hyb}_1$ The simulator $\mathrm{Sim}_{\mathcal{CS}}$ generate a new bijection table as

$$f^* = \begin{pmatrix} 1 & 2 & \dots & n \\ f^*(1) & f^*(2) & \dots & f^*(n) \end{pmatrix}.$$

Then, $\mathrm{Sim}_{\mathcal{CS}}$ permutes the ciphertext $[\![C']\!]$ and sends the disrupted results $[\![C'_{f^*}]\!] = \{[\![c'_{f^*(1)}]\!], [\![c'_{f^*(2)}]\!], ..., [\![c'_{f^*(n)}]\!]\}$ to $\mathcal{U}ser$. The pseudo-randomness Fisher-Yates shuffle algorithm ensures that the order of ciphertext is different. The properties of AHE ensure the indistinguishability between this hybrid with real protocol.

$\mathtt{Hyb}_2$ The $\mathcal{U}ser$ follows the protocol and sends plaintext to the simulator $\mathrm{Sim}_{\mathcal{CS}}$. By executing the Anti-RTP algorithm with f^*, $\mathrm{Sim}_{\mathcal{CS}}$ restore the correct order to get the comparison result of each tree. Therefore, the output of $\mathrm{Sim}^{\pi_{\mathrm{RTPP}}}_{\mathcal{CS}}([\![C']\!], f^*)$ is identically distributed to the view $\mathrm{Real}^{\pi_{\mathrm{RTPP}}}_{\mathcal{CS}}$ of the corrupted cloud server.

The proof of Theorem 2 is completed.

Proof of Theorem 3 We first define the functionality of the GBDT outsourcing inference protocol, denoted as $\mathcal{F}_{GOI}$, as shown in Figure 6.

Input: The $\mathcal{MO}$ holds GBDT $\mathcal{G} = \{\mathcal{T}_1, ..., \mathcal{T}_m\}$. The $\mathcal{U}ser$ holds query sample x, each element in $x \in \mathbb{Z}_N$.
Output: The prediction result $Pr = \sum_{i=1}^{m} \mathcal{T}_i(x)$.

Fig. 6: Functionality of the GBDT outsourcing inference $\mathcal{F}_{GOI}$

Semi-honest model owner security. The simulator $\mathrm{Sim}_{\mathcal{MO}}$ substitutes the original seed with a random number, denoted as $seed^*$, to execute the AHE. Furthermore, $\mathrm{Sim}_{\mathcal{U}}\mathrm{ser}$ generates a fresh seed, denoted as $s^*_{\mathcal{MO}}$, for the PRF. We now show the indistinguishability between real and simulated views by the following hybrid arguments.

$\mathtt{Hyb}_1$ In this hybrid, the simulator $\mathrm{Sim}_{\mathcal{MO}}$ changes the behavior of the model owner. Specifically, for each tree of GBDT, a uniformly random number $seed^*$ is selected to replace the original $seed$ and to perform the AHE encryption. The purpose of the replacement $seed$ is to ensure the correctness of our protocol. The distributions of ciphertext with the same message and pk but different seeds are indistinguishable with semantic security. The simulator cannot distinguish between $seed^*$ and pk-encrypted data and the real $seed$ and pk-encrypted data. In addition, $\mathrm{Sim}_{\mathcal{MO}}$ uniformly sample a new seed $s^*_{\mathcal{MO}} \leftarrow \{0,1\}^\gamma$ to replace the original $s_{\mathcal{MO}}$. The distribution of feature indexes in each tree is indistinguishable because of the pseudorandom function. The IND-CPA secure AHE and PRF ensures that this hybrid possesses indistinguishability from real protocol.

$\mathtt{Hyb}_2$ In this hybrid, the simulator $\mathrm{Sim}_{\mathcal{MO}}$ replaces all encrypted data (e.g., the encrypted threshold $[\![v_{i,j}.\theta]\!]$ and leaf value $[\![v_{i,j}.value]\!]$) sent by the model owner with encrypted data using $seed'$ (e.g., $[\![v_{i,j}.\theta]\!]^*$ and $[\![v_{i,j}.value]\!]^*$). In addition, the simulator computes each feature index using PRF with s'_{MO} (e.g., $v_{i,j}.f'^*$). For the swich flags in internal node $v_{i,j}$, $\mathrm{Sim}_{\mathcal{MO}}$ uniformly generate the random number $v_{i,j}.sf^* \in \{1, -1\}$. Here $*$ indicates a different value for the same input due to the semantic security of the AHE and the property of PRF. Because we just change the seed of AHE and PRF, the properties of AHE and PRF ensure the output of $\mathrm{Sim}^{\pi_{\mathrm{GOI}}}_{\mathcal{MO}}(\mathcal{T}_1, ..., \mathcal{T}_m, seed^*, s^*_{\mathcal{MO}})$ is identically distributed to the view $\mathrm{Real}^{\pi_{\mathrm{GOI}}}_{\mathcal{MO}}$ of the corrupted model owner.

Semi-honest cloud server security. The simulator $\mathrm{Sim}_{\mathcal{CS}}$ perform d rounds of node determination, which consist of π_{SCP} and π_{RTPP}. As we prove in Theorem 1 and 2, the π_{SCP} and π_{RTPP} are secure against semi-honset cloud server $\mathcal{CS}$. $\mathrm{Sim}_{\mathcal{CS}}$ sum the prediction value of leaf nodes as $\sum_{i=1}^{m}[\![\mathcal{T}_i(x))]\!]$. The IND-CPA secure AHE secures the prediction result. Therefore, the output of $\mathrm{Sim}^{\pi_{\mathrm{GOI}}}_{\mathcal{CS}}([\![\mathcal{G}]\!], [\![x]\!], [\![x_1]\!], ..., [\![x_\gamma]\!])$ is identically distributed to the view $\mathrm{Real}^{\pi_{\mathrm{GOI}}}_{\mathcal{CS}}$ of the corrupted cloud server.

Semi-honest user security. The simulator $\mathrm{Sim}_{\mathcal{U}\mathrm{ser}}$ upload the encrypted query sample $[\![x]\!]^*, [\![x_1]\!]^*, ..., [\![x_\gamma]\!]^*$ using $seed^*$ and $s^*_{\mathcal{MO}}$ to the cloud server. Since the π_{SCP} and π_{RTPP} have been proven to be secure for semi-honest users, the output of $\mathrm{Sim}^{\pi_{\mathrm{GOI}}}_{\mathcal{U}\mathrm{ser}}$ is identically distributed to the view $\mathrm{Real}^{\pi_{\mathrm{GOI}}}_{\mathcal{U}\mathrm{ser}}$ of the corrupted user.

The proof of Theorem 3 is completed.

VI. Experiments

A. Experimental Settings

TABLE III: Reference for the UCI datasets

Dataset	Features n	Instances
Nursery	8	12960
Breast-cancer	9	286
Heart-disease	13	303
Adult	14	48842
Credit-screening	15	690
Horse-colic	27	368
Spambase	57	4601

We implement and evaluate our scheme to demonstrate its practicality. All experiments are performed on an Intel Core i7

This article has been accepted for publication in IEEE Transactions on Services Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSC.2024.3395928

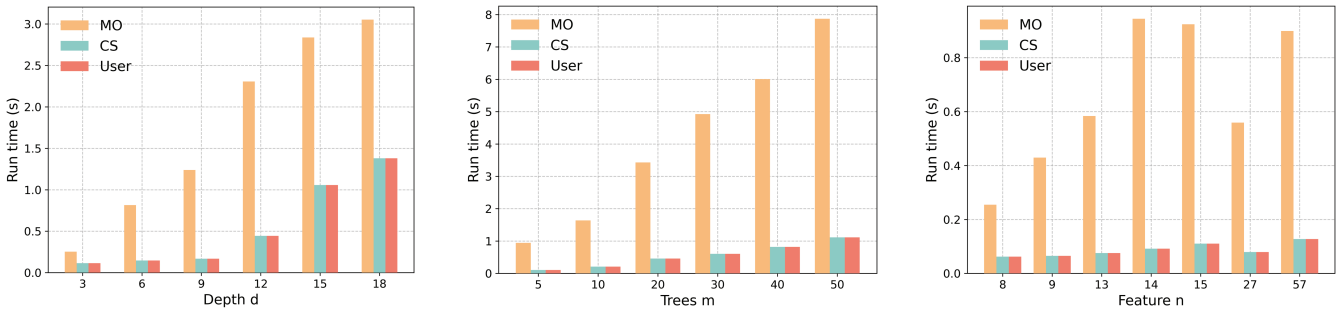

Fig. 7: Computation performance of our scheme for different entities with different d's (m=5, n=14), m's (d=6, n=14), and n's (d=6, m=5)

TABLE IV: Computation performance of the model owner and the cloud server with different d's (m=5, n=14)

Depth d	Model Owner			Cloud Server				
	Trees.trans	Trees.enc	MO.total	SCP	RTP	Anti-RTP	Predict	CS.total
3	0.010567s	0.242599s	0.253166s	0.036839s	0.000038s	0.000009s	0.078177s	0.115063s
6	0.063678s	0.751951s	0.815629s	0.067749s	0.000071s	0.000019s	0.079792s	0.147631s
9	0.116230s	1.121975s	1.238205s	0.088126s	0.000132s	0.000048s	0.079643s	0.167949s
12	0.178506s	2.127499s	2.306005s	0.363499s	0.000175s	0.000082s	0.080423s	0.444179s
15	0.341301s	2.496553s	2.837854s	0.961849s	0.000394s	0.000312s	0.095361s	1.057916s
18	0.529471s	2.522491s	3.051962s	1.267147s	0.000429s	0.000348s	0.112691s	1.380615s

CPU @2.2GHz. Our implementation utilizes Python, with the LightGBM library for classifier generation and the TenSEAL library for CKKS homomorphic encryption operations. In particular, m varies from 5 to 50, d varies from 3 to 18, and n varies from 8 to 57. To minimize errors, each value in the tables is taken from the average of 20 runs of our algorithm. Note that the correctness of our design is guaranteed by the underlying cryptographic primitives. So we focus on the computation and communication performance. The information on datasets is in Table III.

B. Experimental Evaluations

1) ***Computation Performance****:* We examine the computation performance at the MO, the CS, and the $User$, respectively. Our experiments are conducted as follows. (1) Fix (m, n) = (5, 14) and let d varies from 3 to 18. (2) Fix (d, n) = (6, 14) and let m varies from 5 to 50. (3) Fix (d, m) = (6, 5) and let n vary from 8 to 57.

Initially, we conducted experiments to assess the time overhead of each entity under varying parameters, as illustrated in Figure 7. Notably, MO incurs the longest time duration for completing the transformation, which remains within acceptable limits for the offline phase. Indeed, as the depth d increases, the time overhead for $User$ surpasses that of CS. The number of trees m has a greater impact on CS because more trees means more comparisons to be made. Because MO spends most of the time, the time gap between CS and $User$ cannot be clearly seen in Figure 7. Regarding feature n, it significantly impacts the time overhead for all three entities.

Next, We measured the computational performance of each process, and the results are presented in Tables IV, VI, and VII. For the MO task, one only needs to perform private tree transformation on the GBDT after local training. $Trees.trans$ represents the time required for structural transformation of the GBDT, while $Trees.enc$ reflects the computational performance of GBDT encryption. Table IV clearly illustrates that the computation performance of the MO task exhibits a linear increase with respect to the parameter d in GBDT.

TABLE V: Computation performance of user with different d's (m=5, n=14)

Depth d	Data.enc	Middle.dec	Final.dec	Total
3	0.088802s	0.006169s	0.001163s	0.096134s
6	0.135847s	0.012489s	0.001285s	0.149621s
9	0.404622s	0.013436s	0.001139s	0.419197s
12	0.976750s	0.016920s	0.001242s	0.994912s
15	1.320420s	0.017004s	0.001134s	1.338558s
18	2.032602s	0.019281s	0.001156s	2.053039s

Even for the largest GBDT (m=50, d=6, n=14) in our test, the transformation with MO takes only approximately 7.9 seconds. Table VI illustrates that the computation performance of MO increases linearly with the number of trees, denoted as m, in GBDT. From Table VII, we observe a direct relationship between the computational overhead of MO and the variation in n. Overall, the computational performance of MO is more significantly influenced by the values of d and m.

Next, we assess the computational performance of CS, which encompasses both the d-round secure comparison and the permutation of private trees. The term "$Predict$" signifies the time required to sum up the ciphertext predictions from all the trees. In fact, predictions can be computed more quickly for certain trees that reach the leaf nodes earlier in the interaction process. In our analysis, we aggregate all the results after obtaining them, which overestimates the actual overhead. As depicted in Tables IV, VI, and VII, the computational performance of CS is predominantly allocated to the SCP and $Predict$ phases. Moreover, the value of d exerts a substantial influence on the duration of SCP. This is due to the fact that, in the worst-case scenario, each decision tree requires d rounds of comparison to reach a leaf node.

We now examine the computation performance at the $User$

TABLE VI: Computation performance of the model owner and the cloud server with different m's (d=6, n=14)

Trees m	Model Owner			Cloud Server				
	Trees.trans	Trees.enc	MO.total	SCP	RTP	Anti-RTP	Predict	CS.total
5	0.069260s	0.874164s	0.943424s	0.048764s	0.000049s	0.000015s	0.057523s	0.106351s
10	0.107003s	1.529727s	1.636730s	0.097712s	0.000073s	0.000020s	0.110736s	0.208541s
20	0.281701s	3.147459s	3.429160s	0.219759s	0.000098s	0.000044s	0.238164s	0.458065s
30	0.465818s	4.458129s	4.923947s	0.338027s	0.000116s	0.000051s	0.267035s	0.605229s
40	0.610018s	5.395117s	6.005135s	0.464806s	0.000135s	0.000085s	0.354779s	0.819805s
50	0.740381s	7.129192s	7.869573s	0.525133s	0.000187s	0.000142s	0.589709s	1.115171s

TABLE VII: Computation performance of the model owner and the cloud server with different n's (d=6, m=5)

Features n	Model Owner			Cloud Server				
	Trees.trans	Trees.enc	MO.total	SCP	RTP	Anti-RTP	Predict	CS.total
8	0.011145s	0.243765s	0.254910s	0.030492s	0.000039s	0.000011s	0.031790s	0.062332s
9	0.011042s	0.418600s	0.429642s	0.032820s	0.000040s	0.000009s	0.031976s	0.064845s
13	0.024092s	0.559720s	0.583812s	0.036629s	0.000038s	0.000010s	0.038971s	0.075648s
14	0.071103s	0.873225s	0.944328s	0.044743s	0.000052s	0.000012s	0.046828s	0.091635s
15	0.071785s	0.852355s	0.924140s	0.052461s	0.000055s	0.000016s	0.057956s	0.110488s
27	0.038671s	0.520720s	0.559391s	0.037993s	0.000041s	0.000011s	0.041258s	0.079303s
57	0.077138s	0.822025s	0.899163s	0.059844s	0.000066s	0.000015s	0.067425s	0.127350s

TABLE VIII: Computation performance of user with different m's (d=6, n=14)

Trees m	Data.enc	Middle.dec	Final.dec	Total
5	0.131677s	0.012489s	0.001142s	0.145308s
10	0.127917s	0.025373s	0.001298s	0.154588s
20	0.141902s	0.050180s	0.001078s	0.193160s
30	0.132727s	0.076108s	0.001057s	0.209892s
40	0.113075s	0.100655s	0.001130s	0.214860s
50	0.147502s	0.118395s	0.001187s	0.267084s

TABLE IX: Computation performance of user with different n's (d=6, m=5)

Features n	Data.enc	Middle.dec	Final.dec	Total
8	0.075371s	0.006376s	0.001121s	0.082868s
9	0.084374s	0.007187s	0.001172s	0.092733s
13	0.127105s	0.007900s	0.001170s	0.136175s
14	0.136734s	0.009169s	0.001171s	0.147074s
15	0.137129s	0.008483s	0.001301s	0.146913s
27	0.226028s	0.006332s	0.001144s	0.233504s
57	0.369361s	0.009961s	0.001236s	0.380558s

side. Analyzing the experimental results presented in Table V, it becomes evident that the variable d significantly affects the time overhead incurred by the user during data encryption. Increasing the value of m results in higher decryption time for intermediate results, and consequently, the time cost for the user escalates with each increment in m. Even with 50 trees in the GBDT model, the user's total time expenditure remains below 0.27 seconds, as indicated in Table VIII. Lastly, Table IX illustrates the fact that a greater number of features translates to more time required by the user for encryption processes.

*2) **Communication Performance**:* We have evaluated the communication costs for each role, and the corresponding experimental results can be found in Figure 8. First and foremost, we assessed the communication overhead of MO, which exclusively transmits the encrypted GBDT model. Consequently, the communication cost is primarily attributable to the ciphertext of the GBDT model. It is noteworthy that the size of CPTs exhibits minimal sensitivity to variations in d and n, and instead demonstrates a strong correlation with the value of m in GBDT. Even though the number of trees in the model has been altered from 5 to 50, the size of CPTs has increased by a factor of only 4.75.

Next, we examine the communication cost on the CS side. This aspect of communication cost is comprised of two key components: transmitting the ciphertext of disordered node determination results for d rounds ($Middle.cipher$) and the final prediction ciphertext ($Final.cipher$). Notably, $Middle.cipher$ constitutes the primary portion of the communication cost, scaling in tandem with the depth d. Referring to Figure 8, we can derive the following insights. The communication cost on the CS side shows no significant increase with respect to m. The impact of increasing n on the CS remains relatively modest. In the context of the same testing data, the size of the encrypted data remains consistent.

Finally, we evaluate the communication cost for the user, which encompasses both the upload of encrypted prediction data ($Data.encrypted$) and the decryption of intermediate results ($Middle.clean$). As illustrated in Figure 8, the costs vary from 936 KB at d=3 to 2.57 MB at d=18.Even when m is set to 50, the communication cost for the user remains below 3.4 MB, as evidenced in Figure 8. The number of features n has a substantial impact on the user, primarily due to the growth in the volume of encrypted data to be uploaded with increasing values of n.

*3) **Comparison with Prior Works**:* We emphasize that the computation and communication overhead of our scheme for the inference process has significant advantages compared to previous works. Although our performance results are not directly comparable to the prior designs in different scenario settings, it is interesting to observe the potential benefits of our secure outsourcing scheme in Table X.

Specifically, Wu et al. [27] proposed a protocol ($PERF$) using additively homomorphic encryption for privately evaluating decision trees and random forests. In $PERF$, the server needs to transform the decision tree into a complete tree before sending it to the client. However, the computational complexity of the server grows exponentially with the depth of the tree. In contrast, our private tree transformation method randomly expands the leaf nodes without filling up the sparse tree while hiding the node information. Furthermore, Tai et al.

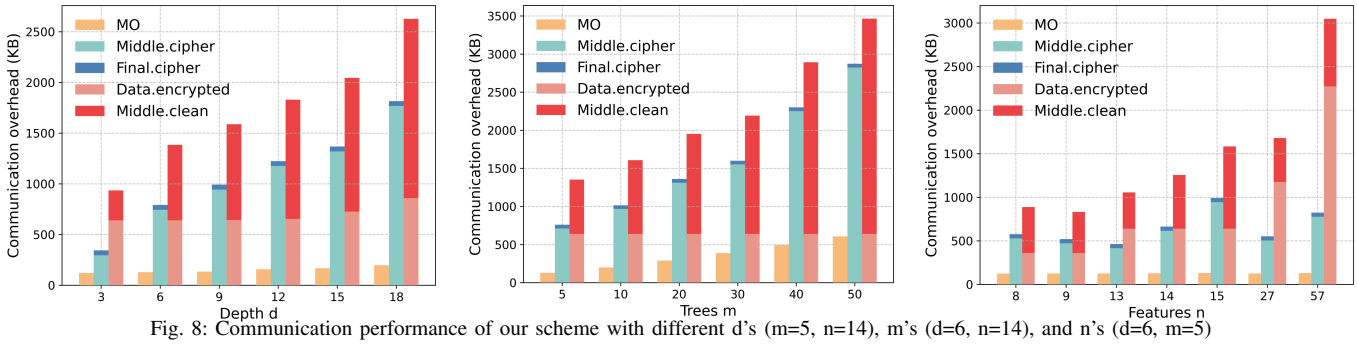

Fig. 8: Communication performance of our scheme with different d's (m=5, n=14), m's (d=6, n=14), and n's (d=6, m=5)

TABLE X: Performance comparison with prior works

(n, d, m)	Scheme	Computation	Communication
(16,10,50)	PERF [27]	117.385s	46.256MB
	Ours	**3.258s**	**9.881MB**
		36.0×	4.7×
(16,12,5)	PPDTE [22]	12.36s	5.06MB
	Ours	**0.61s**	**2.99MB**
		20.3×	1.7×
(57,17,5)	SEODT [18]	6.723s	263.624MB
	Ours	**1.487s**	**3.811MB**
		4.5×	69.2×
(57,17,5)	PDTE [23]	1.969s	17.409MB
	Ours	**1.487s**	**3.811MB**
		1.3×	4.6×

[22] made a significant improvement ($PPDTE$) in efficiency by cleverly exploiting the structure of decision trees. Both our scheme and $PPDTE$ perform well on sparse trees, but their scheme can be very time-consuming for complete trees. Zheng et al. [18] propose a dual-cloud protocol ($SEODT$) using only lightweight cryptography in the online execution of secure inference. Their method is based on the path cost mechanism at the cost of $O(2^d)$ communication complexity for the client. Compared to $SEODT$, our one cloud server setting is more practical. Additionally, our scheme does not traverse all nodes while $SEODT$ needs to compute each internal node during the inference process. At last, Tueno et al [23]'s method ($PDTE$) represents the tree as an array and uses Oblivious RAM (ORAM), which can lead to $d^2 + 3d$ round communication. ORAM has large constants hidden in its asymptotic complexity and a significant setup cost that needs to be amortized over many invocations of the protocol. So the communication overhead of $PDTE$ is higher than our scheme.

VII. Conclusion

In this paper, we propose a novel framework for efficient and secure outsourcing inference based on GBDT. Our approach involves a unique transformation technique for GBDT, complemented by a tailored protocol that leverages pseudorandom function and homomorphic cryptography. This ensures the utmost privacy for the model, test data, and predictions. Experimental results show the effectiveness of our method. In the future, we intend to extend our scheme to the online training process. On the other hand, we also consider doing research on privacy protection for other machine learning algorithms.

References

[1] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of KDD*, 2016, pp. 785–794.

[2] 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," *Proceedings of NIPS*, vol. 30, 2017.

[3] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *Proceedings of NIPS*, vol. 31, 2018.

[4] Y. Xia, C. Liu, Y. Li, and N. Liu, "A boosted decision tree approach using bayesian hyper-parameter optimization for credit scoring," *Expert systems with applications*, vol. 78, pp. 225–241, 2017.

[5] M. Hasan, M. M. Islam, M. I. I. Zarif, and M. Hashem, "Attack and anomaly detection in iot sensors in iot sites using machine learning approaches," *Internet of Things*, vol. 7, p. 100059, 2019.

[6] S. L. Zhu, J. Dong, C. Zhang, Y. B. Huang, and W. Pan, "Application of machine learning in the diagnosis of gastric cancer based on noninvasive characteristics," *Plos one*, vol. 15, no. 12, p. e0244869, 2020.

[7] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *Cryptology ePrint Archive*, 2014.

[8] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. Nascimento, W. S. Poon, and S. Truex, "Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 217–230, 2017.

[9] A. Kiss, M. Naderpour, J. Liu, N. Asokan, and T. Schneider, "Sok: Modular and efficient private decision tree evaluation," *Proceedings on PET*, vol. 2019, no. 2, pp. 187–208, 2019.

[10] 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.

[11] J. Vaidya, B. Shafiq, W. Fan, D. Mehmood, and D. Lorenzi, "A random decision tree framework for privacy-preserving data mining," *IEEE transactions on dependable and secure computing*, vol. 11, no. 5, pp. 399–411, 2013.

[12] L. Zhao, L. Ni, S. Hu, Y. Chen, P. Zhou, F. Xiao, and L. Wu, "Inprivate digging: Enabling tree-based distributed data mining with differential privacy," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2087–2095.

[13] S. Fletcher and M. Z. Islam, "Decision tree classification with differential privacy: A survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, pp. 1–33, 2019.

[14] 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.

[15] Y. Li, Z. L. Jiang, X. Wang, S. M. Yiu, and J. Fang, "Outsourced privacy-preserving random decision tree algorithm under multiple parties for sensor-cloud integration," in *International Conference on Information Security Practice and Experience*. Springer, 2017, pp. 525–538.

[16] M. Joye and F. Salehi, "Private yet efficient decision tree evaluation," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2018, pp. 243–259.

[17] W. j. Lu, Z. Huang, Q. Zhang, Y. Wang, and C. Hong, "Squirrel: A scalable secure two-party computation framework for training gradient boosting decision tree," *Cryptology ePrint Archive*, 2023.

[18] Y. Zheng, H. Duan, C. Wang, R. Wang, and S. Nepal, "Securely and efficiently outsourcing decision tree inference," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[19] J. P. Ma, R. K. Tai, Y. Zhao, and S. S. Chow, "Let's stride blindfolded in a forest: Sublinear multi-client decision trees evaluation." in *Proceedings of NDSS*, 2021.

[20] J. Bai, X. Song, S. Cui, E.-C. Chang, and G. Russello, "Scalable private decision tree evaluation with sublinear communication," in *Proceedings of ASIA CCS*, 2022, pp. 843–857.

[21] K. Sarpatwar, N. K. Ratha, K. Nandakumar, K. Shanmugam, J. T. Rayfield, S. Pankanti, and R. Vaculin, "Privacy enhanced decision tree inference," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 34–35.

[22] R. K. Tai, J. P. Ma, Y. Zhao, and S. S. Chow, "Privacy-preserving decision trees evaluation via linear functions," in *European Symposium on Research in Computer Security*. Springer, 2017, pp. 494–512.

[23] A. Tueno, F. Kerschbaum, and S. Katzenbeisser, "Private evaluation of decision trees using sublinear cost." *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 1, pp. 266–286, 2019.

[24] J. Frery, A. Stoian, R. Bredehoft, L. Montero, C. Kherfallah, B. Chevallier-Mames, and A. Meyre, "Privacy-preserving tree-based inference with fully homomorphic encryption," *arXiv preprint arXiv:2303.01254*, 2023.

[25] R. R. Karn, K. Nawaz, and I. A. M. Elfadel, "Securing decision tree inference using order-preserving cryptography," in *2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2023, pp. 1–5.

[26] Z. Zhu and W. Du, "Understanding privacy risk of publishing decision trees," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2010, pp. 33–48.

[27] D. J. Wu, T. Feng, M. Naehrig, and K. Lauter, "Privately evaluating decision trees and random forests," *Cryptology ePrint Archive*, 2015.

[28] Z. Ma, J. Ma, Y. Miao, and X. Liu, "Privacy-preserving and high-accurate outsourced disease predictor on random forest," *Information Sciences*, vol. 496, pp. 225–241, 2019.

[29] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *arXiv preprint arXiv:2008.06170*, 2020.

[30] 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*. IEEE, 2020, pp. 1–11.

[31] A. C. C. Yao, "How to generate and exchange secrets," in *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 1986, pp. 162–167.

[32] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proceedings of ACM-SIAM symposium on Discrete algorithms*, 2001, pp. 448–457.

[33] L. Li, R. Lu, K. K. R. Choo, A. Datta, and J. Shao, "Privacy-preserving-outsourced association rule mining on vertically partitioned databases," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 8, pp. 1847–1861, 2016.

[34] X. Luo, Y. Jiang, and X. Xiao, "Feature inference attack on shapley values," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 2233–2247.

[35] Y. Aono, T. Hayashi, L. Wang, S. Moriai *et al.*, "Privacy-preserving deep learning via additively homomorphic encryption," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1333–1345, 2017.

[36] R. A. Fisher and F. Yates, *Statistical tables for biological, agricultural and medical research*. Hafner Publishing Company, 1953.

[37] R. Durstenfeld, "Algorithm 235: random permutation," *Communications of the ACM*, vol. 7, no. 7, p. 420, 1964.

[38] G. Xu, H. Li, S. Liu, K. Yang, and X. Lin, "Verifynet: Secure and verifiable federated learning," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 911–926, 2019.

[39] G. Xu, H. Li, Y. Zhang, S. Xu, J. Ning, and R. H. Deng, "Privacy-preserving federated deep learning with irregular users," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 1364–1381, 2020.

Shuai Yuan Shuai Yuan received a B.S. degree in information security from the University of Science and Technology Beijing, and he is currently working toward a Ph.D. degree in cyber security at the University of Electronic Science and Technology of China. His research interests include applied cryptography and privacy-preserving machine learning.

Hongwei Li Hongwei Li (Fellow, IEEE) is currently the Associate Dean at School of Computer Science and Engineering, University of Electronic Science and Technology of China. He received the Ph.D. degree from University of Electronic Science and Technology of China in June 2008. He worked as a Postdoctoral Fellow at the University of Waterloo from October 2011 to October 2012. His research interests include network security and applied cryptography. Dr. Li has published more than 100 technical papers. He is the sole author of a book, Enabling Secure and Privacy Preserving Communications in Smart Grids (Springer, 2014). Dr. Li serves as the Associate Editors of IEEE Internet of Things Journal, and Peer-to-Peer Networking and Applications, the Guest Editors of IEEE Network, IEEE Internet of Things Journal and IEEE Transactions on Vehicular Technology. He also served the Technical Symposium Co-chairs of IEEE ICC 2022, ACM TUR-C 2019, IEEE ICCC 2016, IEEE GLOBECOM 2015 and IEEE BigDataService 2015, and Technical Program Committees for many international conferences, such as IEEE INFOCOM, IEEE ICC, IEEE GLOBECOM, IEEE WCNC, IEEE SmartGridComm, BODYNETS and IEEE DASC. He won Best Paper Awards from IEEE ICPADS 2020 and IEEE HEALTHCOM 2015. Dr. Li currently serves as the Vice Chair(conference) of IEEE ComSoc CIS-TC. He is the Fellow of IEEE and the Distinguished Lecturer of IEEE Vehicular Technology Society.

Xinyuan Qian Xinyuan Qian received B.Eng. degree in security engineering in People's Public Security University of China in 2018 and M.Eng. degree in computer technology in University of Chinese Academy of Sciences in 2021. He is currenting pursuing the Ph.D. degree in School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include IBE, ABE, FE, FHE, applied cryptography, and privacy-preserving machine learning.

Meng Hao Meng Hao received the B.S. degree in information security in 2018, from the University of Electronic Science and Technology of China (UESTC). He is currently working toward the Ph.D. degree in cyber security at UESTC. His research interests include applied cryptography and privacy-preserving deep learning.

Yixiao Zhai Yixiao Zhai received her B.S. degree in Information Security from the University of Electronic Science and Technology of China (UESTC) in 2020. She received M.S. degree in 2023 from the UESTC. Her research interests include adversarial attacks and privacy-preserving deep learning.

Guowen Xu Guowen Xu is currently a Postdoc at City University of Hong Kong under the supervision of Prof. Yuguang Fang. He was a Research Fellow at Nanyang Technolgoical University from March 2021 to May 2023. He received his Ph.D. degree in 2020 from the University of Electronic Science and Technology of China. His research interests include applied cryptography and privacy-preserving deep learning.