ELXGB: An Efficient and Privacy-Preserving XGBoost for Vertical Federated Learning

Wei Xu, Hui Zhu*, *Senior Member, IEEE*, Yandong Zheng, Fengwei Wang, *Member, IEEE*, Jiaqi Zhao, Zhe Liu, *Senior Member, IEEE*, and Hui Li, *Member, IEEE*

Abstract—With the rapid growth of Internet data volumes, big data analysis technologies have gradually permeated all aspects of life. However, the existence of data silos and the promulgation of relevant regulations make it challenging to apply these technologies. In this context, federated learning provides a feasible solution. Especially, XGBoost schemes for vertical federated learning have attracted much attention due to the widespread use of XGBoost. However, these schemes have limitations in terms of security or efficiency. To address these issues, we propose an efficient and privacy-preserving vertical federated learning framework based on the XGBoost algorithm, namely ELXGB, which achieves secure data alignment, XGboost training, and inference services. First, we design two node split algorithms based on homomorphic encryption and differential privacy, which securely and efficiently achieve tree node generation to construct the global model. Then, we utilize attribute obfuscation and direction obfuscation to achieve a secure inference algorithm, which avoids sensitive information leakage and protects the global model. Additionally, the global model of ELXGB is designed to be centralized, which does not require all participants to stay online for inference. Detailed security analysis demonstrates that ELXGB is privacy-preserving. Moreover, extensive experiments on real-world datasets indicate that ELXGB achieves high efficiency without sacrificing model accuracy.

Index Terms—Vertical Federated Learning, Homomorphic Encryption, Differential Privacy, Privacy-Preserving, XGBoost

1 Introduction

With the increasing awareness of data privacy protection, the application of big data analysis technologies has been affected. The regulations like the General Data Protection Regulation (GDPR) [1] clearly state that processors should strengthen their obligations to protect data privacy. Due to restrictions on data usage, Internet data cannot be easily exchanged among enterprises. Moreover, model training based on local data usually suffers from a limited number of data samples, which causes the quality of the trained model to be poor. Consequently, Google introduced a state-of-the-art framework for federated learning (FL) [2], which provides a feasible solution for the above problems.

Federated learning is a distributed learning method where many participants cooperatively train a global model under a service provider's leadership, while participants' data remains on the local side. It is well acknowledged that there are three kinds of mainstream federated learning frameworks [3], including horizontal federated learning, vertical federated learning, and federated transfer learning. Under these frameworks, many federated learning schemes have been proposed by utilizing traditional machine learning algorithms. Especially, the XGBoost schemes [4] for vertical federated learning are particularly popular due to the wide application of XGBoost. They aim to achieve gradient aggregation to find the best node split. Existing XGBoost schemes solve the above problem based on the security techniques of homomorphic encryption (HE), secure multi-party computation (SMC), and differential privacy (DP). However, they suffer from the limitations of efficiency or security [5].

Specifically, schemes [6]–[8] utilize homomorphic encryption [9] to train a high-quality boosting tree model. However, the encryption overhead might not be tolerable in the massive data scenario. Moreover, the intermediate information like data labels and instance indexes might be obtained by model inversion attacks [10]. To alleviate intermediate information leakage, schemes [11]–[13] present frameworks based on secret sharing [14] and secure multi-party computation [15] to build a secure XGBoost model with vertically partitioned data. All participants train the global model by data slices from others, which protects the data labels and data distributions well. However, their global model is distributed among various participants, and each participant just owns a part of the model. When providing the inference, all participants must keep synchronized and online to perform the services for each user, which results in the distributed framework being used to the maximum extent and increasing communication overhead for users.

To solve the above problems, the scheme [16] builds the global model based on secret sharing [17] and function encryption [18] between the cloud service provider (CSP) and one participant. Since all participants precompute the intermediate information, divide data into two parts, and send them to CSP and the participant, the framework has better

- *W. Xu, H. Zhu, Y. Zheng, F. Wang, J. Zhao and H. Li are with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, 710071, China (e-mail: xuwei_1@stu.xidian.edu.cn, zhuhui@xidian.edu.cn, zhengyandong@xidian.edu.cn, wangfengwei@xidian.edu.cn, jq_zhao@stu.xidian.edu.cn, and lihui@mail.xidian.edu.cn) (Corresponding author: Hui Zhu).*
- *Z. Liu is with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 210016, China, and also with the Zhejiang Laboratory, Hangzhou, Zhejiang 311100, China (e-mail: zhe.liu@nuaa.edu.cn).*

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.2024.3394706

performance in terms of efficiency. However, the global model is leaked to CSP, which increases the risk of sensitive data leakage [19]. Moreover, the inference process needs to traverse every node of the tree model, which generates huge overhead. To further protect the model, schemes [7], [11] also utilize differential privacy [20], [21] to prevent data labels and leaf weight leakage. However, they lead to the loss of model accuracy to some degree. Moreover, almost all schemes leak the tree node information, which might leak users' sensitive information by property inference attacks [22] during the inference. Therefore, these problems motivate us to study an efficient and privacy-preserving XGBoost for a vertical federated learning framework.

To tackle the above problems, we propose an efficient and privacy-preserving XGBoost framework for vertical federated learning named ELXGB. First, based on Paillier homomorphic encryption and differential privacy, we design two node split algorithms, which securely and efficiently achieve tree node generation to construct the global model. Then, by attribute obfuscation and direction obfuscation, we design a secure and accurate inference algorithm to avoid sensitive information leakage and protect the global model. Overall, our main contributions are the following threefold.

• *First, ELXGB achieves the secure XGBoost training and inference.* We design two node split algorithms that provide a secure training process and ensure the confidentiality of model weight and data order. Moreover, we propose two obfuscation algorithms to prevent sensitive information leakage during the inference, which protect node's attribute information and the global model. Security analysis shows that ELXGB is privacy-preserving.

• *Second, ELXGB guarantees a high-accuracy boosting tree model, which almost achieves the same accuracy as the XGBoost.* ELXGB provides various services, including data alignment, training, and secure prediction. We apply HE to finish the first tree built and use DP to finish other trees built during the training. Moreover, the detailed design makes the noise have little influence on the model. The theoretical analysis and extensive experimental results demonstrate that ELXGB achieves almost the same high accuracy as XGBoost.

• *Third, ELXGB is efficient for model training and inference.* During the training process, we utilize HE and DP to finish the training efficiently. The expensive overhead is only concentrated on establishing the first node of the first tree. Moreover, to provide more efficient inference service, the global model is securely constructed in the participant who owns data labels. It is unnecessary to request all participants to service users in some scenarios.

The rest of this paper is organized as follows. Section 2 describes the system model, threat model, and design goals. Section 3 discusses preliminaries of our scheme. Section 4 overviews our goals and algorithm designs. Section 5 proposes the building blocks. Section 6 introduces our scheme. Section 7 analyzes the security of our scheme. Section 8 evaluates various performance aspects of the proposed framework. Section 9 introduces some of the closely related schemes. Section 10 discusses here security and optimization extensions. Finally, Section 11 concludes the paper.

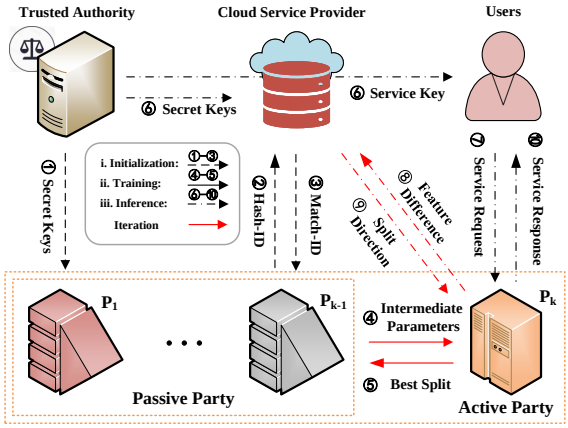

Fig. 1. System model under consideration

2 Problem Formulation

In this section, we formally describe our system model and threat model, and then identify our design goals.

2.1 System Model

Our model mainly focuses on designing an efficient and privacy-preserving XGBoost framework for vertical federated learning, which involves four types of entities (a trusted authority TA, a cloud service provider CSP, multiple participants $\{P_1, \cdots, P_K\}$ and users), as shown in Fig. 1. The active party P_K, who wants to obtain an accurate model, initiates a crowd-sourced learning task, publishes the corresponding rewards on a TA, and calls for contributions from the public. The interested participants $\{P_k\}_1^{K-1}$, also called the passive parties, submit their respective attributes to TA, and TA marks each attribute with a serial number, which will be sent to the corresponding parties. Moreover, all participants have vertically distributed data, and P_K also has the data labels. The participants are large organizations and they do not drop out during the training or inference. After finishing these consensuses, the system starts the initialization. First, TA generates secret keys and distributes corresponding keys to participants. Then, all participants match their data with the assistance of CSP for model training. During the training, P_K iteratively trains the global model with $\{P_k\}_1^{K-1}$. To get a better user experience from P_K, the users ask a request for service keys from TA. Moreover, CSP also receives the corresponding key to assist P_K to design a personal service for the users.

2.2 Threat Model

In our threat model, we consider that participants and CSP are honest-but-curious [23]. And we assume that participants do not collude with CSP due to the conflict of interest. Under these assumptions, ELXGB should ensure private data security and resist the following threat model. First, during data alignment, CSP honestly executes Private Set Intersection (PSI). However, it might utilize received information to infer sensitive attributes of all participants (e.g., name, wohnort, hobby, etc). Then, during the training

process, participants collaboratively train the boosting tree model under the leadership of P_K, where $\{P_k\}_1^{K-1}$ are curious about data labels of P_K and P_K is also craving for inferring data characteristics of $\{P_k\}_1^{K-1}$, such as data order and instance distributions. Especially, $\{P_k\}_1^{K-1}$ attempt to utilize interactive information to infer P_K's sensitive information by model inversion attacks [10] [19]. During the inference, CSP and P_K might utilize the tree nodes attributes to obtain the sensitive information of $\{P_k\}_1^{K-1}$ and users by property inference attacks [22]. Moreover, CSP is also curious about the global model, and it attempts to steal the global model. In addition, it is worth noting that there may be some other active attacks (e.g., poisoning attack [24], backdoor attack [25], etc.) in FL. Since our scheme mainly focuses on achieving privacy-preserving boosting tree model training and inference, these attacks are currently out of the scope of this paper and will be considered in future work.

2.3 Design Goals

Based on the system and threat models mentioned above, in our scheme, we aim to achieve an efficient and privacy-preserving XGBoost framework for vertical federated learning under the premise of ensuring security. Specifically, ELXGB should satisfy the following two objectives.

• **Security.** Under the honest-but-curious assumptions, CSP, the active party, and the passive parties might be interested in information from each other. Therefore, ELXGB needs to protect the privacy of ID, data labels, data characteristics, attributes information, and global model.

• **Efficiency.** When training a large scale of data, security techniques such as HE used in XGBoost model will incur huge overhead. Therefore, ELXGB needs to design efficient algorithms to achieve data alignment, model training, and inference service, as shown in Fig. 1.

3 Preliminaries

In this section, we briefly review one of the most widely used boosting tree models - XGBoost, Paillier cryptosystem, and differential privacy, which serve as the basis of our scheme. For a clear description, the commonly used notations are listed in TABLE 1.

TABLE 1
Notations In Model

Notations	Definition
K, N_i, N, M	The numbers of participants, data samples from the ith participant, algined data and data attributes.
ssk	Symmetric key for data alignment.
(P_{KT}, S_{KT})	A pair of Paillier's keys for training.
(P_{KS}, S_{KS})	A pair of Paillier's keys for inference.
(ϵ, δ)	Differential Privacy' algorithm parameters.
g, h	Gradient and hessian in XGBoost.
$[\![g_i]\!], [\![h_i]\!]$	The ith encrypted g and h by Paillier
$\tilde{g}_i, \tilde{h}_i$	The ith noised g and h by DP
$\mathcal{B}, I$	$\mathcal{B}$ represents Bucket and I represents bits. They are information about if data in Bucket.
eps	The number of I in each $\mathcal{B}$ is $\frac{1}{eps}$.
$\triangle$	The scale factor.
T, D	Max tree number and max depth in model.
w_i^t	Weight of the i-th data in t-th tree.

3.1 XGBoost

XGBoost [4] is an efficient and widely used boosting tree algorithm, which integrates many weak classifiers, such as CART [26], to form a strong classifier. Given a dataset $\mathbb{D} = \{(X_i, y_i)\}(|\mathbb{D}| = n, X_i \in \mathbb{R}^m, y_i \in \mathbb{R})$ with n samples and m attributes, XGBoost predicts the i-th instance by using T weak classifiers as follows.

$$\hat{y}_i = \sum_{t=1}^{T} f_t(X_i) = \sum_{t=1}^{T} w_i \tag{1}$$

Each $f_t(\cdot)$ is a decision tree and $f_t(X_i)$ maps X_i to the corresponding leaf weight w_i of $f_t(X_i)$. $f_t(\cdot)_{t:1\le t\le T}$ are constructed one by one. When $f_t(\cdot)$ is constructed, the objective function $\mathcal{L}^{(t)}$ can be defined as

$$\mathcal{L}^{(t)} = \sum_{i}^{n} l(y_i, \hat{y}_i^t) + \Omega(f_t)$$

where $\Omega(f) = \gamma\Gamma + \frac{1}{2}\lambda\|W\|^2$, and l is the loss function. Ω is a regularization item, where Γ is the number of leaf nodes in $f_t(\cdot)$ and W is the set of weight values in $f_t(\cdot)$'s leaf nodes. The objective function of model can be optimized based on the second-order Taylor expansion.

$$\mathcal{L}^{(t)} \simeq \sum_{i}^{n}[l(y_i, \hat{y}_i^{(t-1)}) + g_i^t f_t(X_i) + \frac{1}{2}h_i^t f_t^2(X_i)] + \Omega(f_t) \tag{2}$$

where $g_i^t = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, $h_i^t = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$. g_i^t and h_i^t are the first and second derivatives of the loss function respectively on t-th tree. Minimizing the objective function above is equivalent to maximizing the gain of each node [4]. Therefore, to construct the t-th tree, the model needs to split nodes so that each node maximizes the gain G_I as follows

$$G_I = \frac{1}{2}[\frac{(\sum_{i\in I_L} g_i^t)^2}{\sum_{i\in I_L} h_i^t + \lambda} + \frac{(\sum_{i\in I_R} g_i^t)^2}{\sum_{i\in I_R} h_i^t + \lambda} - \frac{(\sum_{i\in I} g_i^t)^2}{\sum_{i\in I} h_i^t + \lambda}] - \gamma \tag{3}$$

until it reaches the maximum tree depth, where I is the sample index set in a node and $I = I_R + I_L$. By sorting each column attribute in advance, I can be divided into different sets. At the same time, the gradients g_i^t and h_i^t are counted through the indexes to obtain the maximum gain of each node. When the optimal tree structure is built, the optimal weight w_j^* of the leaf node can be computed by the following equation, where I_j is the sample index set of the node.

$$w_j^* = w_{i\in I_j} = -\frac{\sum_{i\in I_j} g_i^t}{\sum_{i\in I_j} h_i^t + \lambda} \tag{4}$$

3.2 Paillier

As one of the most popular homomorphic encryption schemes, Paillier cryptosystem [9] is widely used in federated learning [27], [28]. It can be described as follows.

• $\mathsf{Paillier.KeyGen}(\kappa)$: Given a security parameter κ and two big prime numbers $\mathtt{p}, \mathtt{q}$ with $|\mathtt{p}| = |\mathtt{q}| = \kappa$, compute $\mathtt{n} = \mathtt{pq}$, $\lambda = lcm(\mathtt{p}-1, \mathtt{q}-1)$, where lcm is the lowest

common multiple. Then, select a random number $\mathtt{g} \in \mathcal{Z}^*_{\mathtt{n}^2}$, satisfying $\gcd(\mathtt{L}(\mathtt{g}^\lambda \bmod \mathtt{n}^2), \mathtt{n}) = 1$, where $\mathtt{L}(x) = \frac{\mathtt{x}-1}{\mathtt{n}}$. Next, compute $\mu = \mathtt{L}(\mathtt{g}^\lambda \bmod \mathtt{n}^2)^{-1} \bmod \mathtt{n}$. The public key is $mpk = (\mathtt{n}, \mathtt{g})$ and the secret key is $msk = (\lambda, \mu)$.

• Paillier.Enc$(mpk, \mathtt{m})$: For any message $\mathtt{m} \in \mathcal{Z}_\mathtt{n}$, select a random number $\mathtt{r} \in \mathcal{Z}^*_\mathtt{n}$ and the ciphertext can be calculated with mpk as $\mathtt{c} = \mathtt{g}^\mathtt{m} \cdot \mathtt{r}^\mathtt{n} \bmod \mathtt{n}^2$.

• Paillier.Dec$(msk, \mathtt{c})$: For any ciphertext $\mathtt{c} \in \mathcal{Z}^*_{\mathtt{n}^2}$, the plaintext can be retrieved with the secret key msk through computing $m = \mathtt{L}(\mathtt{c}^\lambda \bmod \mathtt{n}^2) \cdot \mu \bmod \mathtt{n}$.

The additive homomorphism of Paillier means: any two ciphertexts $[\![m_1]\!]$ and $[\![m_2]\!]$ satisfy $[\![m_1]\!] \cdot [\![m_2]\!] \rightarrow [\![m_1 + m_2]\!]$.

The scalar multiplication of Paillier means: a plaintext m_1 and a ciphertext $[\![m_2]\!]$ satisfy $[\![m_2]\!]^{m_1} \rightarrow [\![m_1 \cdot m_2]\!]$.

3.3 Differential Privacy

Differential privacy (DP) is an efficient and privacy-preserving technique. This technique can get the invariable nature of statistical analysis while sacrificing certain accuracy to ensure data security. The definition of differential privacy [29] is as follows:

Definition 1. *((ϵ, δ)-differential privacy.) Given a randomized algorithm $\mathcal{A}$, result$(\mathcal{A})$ represents the set composed of all outputs of the algorithm. For any two datasets $\mathcal{X}$ and $\mathcal{Y}$, if the two differ by one record, any subset $\mathcal{S}$ on result$(\mathcal{A})$ satisfies*

$$Pr[\mathcal{A}(\mathcal{X}) \in \mathcal{S}] \leq e^\epsilon \cdot Pr[\mathcal{A}(\mathcal{Y}) \in \mathcal{S}] + \delta$$

where ϵ expresses privacy parameter and δ represents failure probability. When $\delta = 0$, it can be considered ϵ-differential privacy [30].

Definition 2. *(Sensitivity.) Given a function $\mathcal{A}$: $\mathcal{X} \rightarrow \mathcal{R}^d$,$\mathcal{Y} \rightarrow \mathcal{R}^d$, the sensitivity of function $\mathcal{A}$ is as follows:*

$$\Delta\mathcal{A} = \max_{\mathcal{X},\mathcal{Y}} \parallel \mathcal{A}(\mathcal{X}) - \mathcal{A}(\mathcal{Y}) \parallel_1 .$$

The sensitivity of the function reflects the maximum possible change due to the addition or removal of a single sample.

4 Overview

In this section, we describe our goals and algorithm designs from a high level. In the vertical federated XGBoost scheme, there is one active party P_K and $K-1$ passive parties $\{P_k\}_1^{K-1}$, and each party owns the data set $\mathbb{D}_{i:1\leq i\leq K}$ locally. Moreover, P_K has the data label set Y. The system is to maximize each node's gain of the XGBoost model through federated learning. Assuming that its task is a classification task, the gradients can be calculated as $g_i = y_{i,pred} - y_i$ and $h_i = y_{i,pred} \cdot (1 - y_{i,pred})$ by P_K, and $y_{i,pred}$ is defined:

$$y_{i,pred} = -\frac{1}{1+e^{-\hat{y}_i}} = -\frac{1}{1+e^{-\sum_{t=1}^{T} w_i^t}}$$

Then, the problem can be described as follows.

To find the best split, P_K needs to calculate the maximum gain as Eq. (3) of each node from all attributes. The gain can be further decomposed into calculating the sums $\sum_{I_L} g_i$, $\sum_{I_R} g_i$ and $\sum_I g_i$ (resp. h_i). Then, I_L (resp. I_R) with the maximum gain will be used as I for the next left (resp. right) node split. During the process, these sample index sets I from the passive parties involve their data distribution, and the gradients from the active party involve the data labels. There are security needs to aggregate gradients by the samples, which would not disclose sensitive information to each other. Moreover, it is also challenging to balance efficiency and security.

Therefore, to achieve an efficient and privacy-preserving XGBoost scheme for vertical federated learning, we design and finish our goals from the following three aspects.

• *Goal 1: How to efficiently train the global model independently of the model structure while ensuring security?* In almost schemes, although they can finish the model training, they still suffer from the limitations of efficiency and security. Moreover, these schemes will generate more overhead as the model structure becomes richer. Therefore, we propose two node split algorithm: HE-based node split (HENS) and DP-baesd node split (DPNS), to finish the model training. We utilize HENS to construct the first tree and utilize DPNS to construct other trees. The expensive overhead is only concentrated on establishing the first node of the first tree. Moreover, the gradient cache and gradient recovery mechanisms are used to ensure the model accuracy.

• *Goal 2: How to securely construct the global model to provide efficient inference services?* Almost privacy-preserving XGBoost schemes construct distributed models among all participants, which might protect each node information of the global models. However, the communication cost of users is greatly increased during the inference, which makes these schemes inflexible. Constructing the global model in one participant provides a feasible solution, but it also brings new problems. Exposure of node's attribute information might leak split threshold [22]. Therefore, we design an attribute obfuscation algorithm to hide the node information, which allows us to construct the global model on P_K. Moreover, it also protects the split threshold of nodes and data distributions of $\{P_k\}_1^{K-1}$.

• *Goal 3: How to protect the data privacy during the inference?* Based on *Challenge 2*, if the secure global model is built in P_K, P_K needs to finish the inference for users. To reduce the user's communication overhead, CSP assists users to finish the inference. However, the sensitive information of users and the global model might be leaked during the process. Therefore, the attribute obfuscation algorithm is used to protect the sensitive information of users. Moreover, we propose a direction obfuscation algorithm to protect the split directions and the global model.

5 Building Blocks

In this section, we first introduce a bucket generation and update method. Then, we present an HE-based node split algorithm (HENS) and a DP-based node split algorithm (DPNS), which are building blocks of our scheme.

5.1 Bucket Generation and Update

One of the key problems in tree learning models is to find the best split by Eq. (3). Therefore, to get I_L, I_R, and I, XGBoost [4] provides two approximate approaches to update the bucket: the local proposal and the global proposal.

First, the bucket generation divides the values of each attribute into multiple buckets. Each bucket is denoted by $\mathcal{B}$ and corresponds to $\frac{1}{eps}$ N-dimensional binary vectors. Each

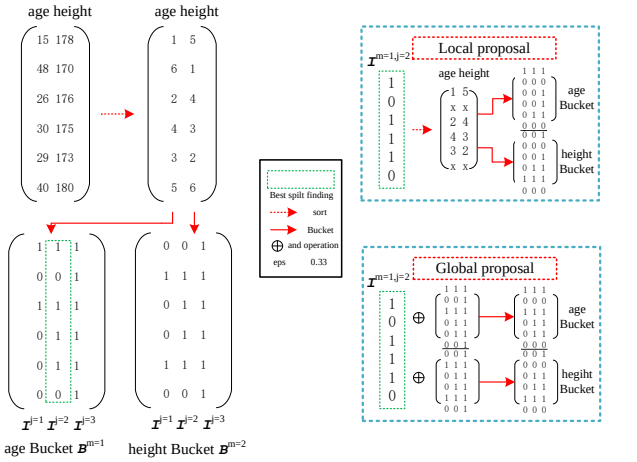

Fig. 2. Example about the local/global buckets division methods. The left represents bucket $\mathcal{B}$ about the first split, and the right represents $\mathcal{B}$ about the next left node split. The local proposal in the up-right side, and the global proposal is in the bottom-right side.

Algorithm 1 HE-based Node Split Algorithm (HENS)

▷ *used to construct d-th node in the first tree* ◁
$P_K \xrightarrow{[\![g^1]\!],[\![h^1]\!]} \{P_k\}_1^{K-1}$.
▷ *Internal Nodes Building* ◁
$P_K \xrightarrow{I^{d,1}} \{P_k\}_1^{K-1}$, and $\{P_k\}_1^K$ generate $\mathcal{B}^{d,1}$ by $I^{d,1}$.
P_K starts the clock.
$\{P_k\}_1^{K-1} \xrightarrow{<[\![g_{I_L}]\!]>^{d,1},<[\![h_{I_L}]\!]>^{d,1}} P_K$.
P_K ends the clock.
P_K decrypts the received ciphertexts.
P_K calculates the maximum gain $G_I^{d,1}$.
if $G_I^{d,1}$ in P_K **then**
  P_K sets node information $\{\mathbf{I}_L^{m,j,d,1}, \mathcal{T}_0^{d,1}\}$.
else
  ▷ *the participant $\mathbf{P}_I$ who exists max gain* ◁
  $P_K \xrightarrow{\text{Request}} \mathbf{P}_I$.
  $\mathbf{P}_I \xrightarrow{\{\mathbf{I}_L^{m,j,d,1}, \mathcal{T}_0^{d,1}\}} P_K$.
  P_K generates model.
▷ *Leaf Nodes Building* ◁
P_K generates the leaf weight.

vector is denoted by I, where $I[j] = 1$ if the corresponding data is in the vector and $I[j] = 0$ otherwise for $1 \le j \le N$. Then, the local and global proposal are utilized to update the bucket. The local proposal prefers accuracy, meaning participants must reorder every update's attribute value. The global proposal prefers efficiency, meaning participants can prepare the buckets for finding the best split in advance. The global proposal with more $\frac{1}{eps}$ can also reach the same accuracy as the local proposal [4]. An example about the local/global buckets division methods is shown in Fig. 2. In our scheme, we use the global proposal for efficiency. Of course, our scheme is also suitable for the local proposal.

5.2 HE-based Node Split Algorithm (HENS)

The HENS algorithm utilizes Paillier homomorphic encryption to privately achieve the tree node generation. Especially, internal nodes and leaf nodes will be built in different ways as shown below.

1) *Internal Nodes Building*

The goal of an internal node building is to find the best split attribute and the split value in the current node. Detailed steps of the algorithm is introduced as below.

Step 1: Pre-preparation

First, P_K calculates the predicted results and gradients using the loss function, and maintains accuracy by a scale factor $\triangle$ as

$$\begin{aligned} y_{i,pred}^1 &= -\frac{1}{1+e^{-w_i^0}}, 1 \le i \le N \\ [\![g_{i:1\le i\le N}^1]\!] &= [\![\triangle \cdot (y_{i,pred}^1 - y_i)]\!] \\ [\![h_{i:1\le i\le N}^1]\!] &= [\![\triangle \cdot y_{i,pred}^1 (1 - y_{i,pred}^1)]\!]. \end{aligned} \tag{5}$$

Then, P_K sends gradients to $\{P_k\}_1^{K-1}$. In addition, to construct the first node of the first tree, P_K initializes an N-dimensional vector $I^{d=1,1} = [1, 1, \cdots, 1]$ and sends it to $\{P_k\}_1^{K-1}$.

Step 2: Best split finding

On receiving $I^{d,1}$, $\{P_k\}_1^K$ utilizes $I^{d,1}$ to update its bucket $\mathcal{B}$ as shown in Fig. 2. After that, to obtain the gain on the d-th node of the first tree about the j-th I of the m-th $\mathcal{B}$, $\{P_k\}_{k=1}^{K-1}$ computes

$$[\![\sum_{I_L} g_i^1]\!]^{m,j,d,1} = [\![g_1^1]\!]^{I_1^{m,j,d,1}} \cdot [\![g_2^1]\!]^{I_2^{m,j,d,1}} \cdots [\![g_N^1]\!]^{I_N^{m,j,d,1}}$$
$$[\![\sum_{I_L} h_i^1]\!]^{m,j,d,1} = [\![h_1^1]\!]^{I_1^{m,j,d,1}} \cdot [\![h_2^1]\!]^{I_2^{m,j,d,1}} \cdots [\![h_N^1]\!]^{I_N^{m,j,d,1}}.$$

Let $<[\![g_{I_L}]\!]>^{d,1} = \{\{[\![\sum_{I_L} g_i]\!]^{m,j,d,1}\}_{m=1}^{M-M_{P_K}}\}_{j=1}^{\frac{1}{eps}}$ and $<[\![h_{I_L}]\!]>^{d,1} = \{\{[\![\sum_{I_L} h_i]\!]^{m,j,d,1}\}_{m=1}^{M-M_{P_K}}\}_{j=1}^{\frac{1}{eps}}$. Then, $\{P_k\}_1^{K-1}$ send $<[\![g_{I_L}]\!]>^{d,1}$ and $<[\![h_{I_L}]\!]>^{d,1}$ to P_K. To avoid wasting time to wait the passive participants with slow connections, P_K can choose to start the clock after collecting the first encrypted value. After a decent interval, P_K rejects to receive the values. The clock time can be adjusted according to the experience. Then, P_K decrypts the received $<[\![g_{I_L}]\!]>^{d,1}$ and $<[\![h_{I_L}]\!]>^{d,1}$, and calculates $\sum_{I_L} g_i^{m,j,d,1}$. It continues

$$\sum_{I_R} g_i^{m,j,d,1} = \sum_{I} g_i^{m,j,d,1} - \sum_{I_L} g_i^{m,j,d,1}$$

and computes the gain $G_I^{m,j,d,1}$ as Eq. (3). Then, it computes

$$G_I^{d,1} = \max\{G_I^{m,j,d,1} | 1 \le m \le M, 1 \le j \le \frac{1}{eps}\}.$$

If $G_I^{d,1}$ is from $\{P_k\}_1^{K-1}$, P_K sends $\{m, j\}$ to $\mathbf{P}_I$ who owns $G_I^{d,1}$, and requests $\mathbf{P}_I$ to build a tree node.

Step 3: Node construction based on attribute obfuscation

On receiving $\{m, j\}$, $\mathbf{P}_I$ identifies the vector $\mathbf{I}_L^{m,j,d,1}$ and the split threshold $\tau^{d,1}$. Constructing a tree node requires not only $\{\mathbf{I}_L^{m,j,d,1}, \tau^{d,1}\}$ but also the attribute. However, the exposure of attributes increases the probability of threshold leakage by property inference attacks [22]. To improve the

Algorithm 2 DP-based Node Split Algorithm (DPNS)

▷ *used to construct d-th node in $t_{:t\geq 2}$-th tree* ◁
P_K calculates g^t and h^t.
P_K calculates $\tilde{g}^t$ and $\tilde{h}^t$, and $P_K \xrightarrow{\tilde{g}^t,\tilde{h}^t} \{P_k\}_1^{K-1}$.
▷ *Internal Nodes Building* ◁
$P_K \xrightarrow{I^{d,t}} \{P_k\}_1^{K-1}$, and $\{P_k\}_1^K$ generate $\mathcal{B}^{d,t}$ by $I^{d,t}$.
P_K starts the clock.
$\{P_k\}_1^K$ calculates the local maximum gain $G_I^{k,d,t}$.
$\{P_k\}_1^{K-1} \xrightarrow{G_I^{k,d,t}} P_K$.
P_K ends the clock.
P_K calculates the global maximum gain $G_I^{d,t}$.
if $G_I^{d,t}$ in P_K **then**
  P_K sets node information $\{\mathbf{I}_L^{m,j,d,1}, \mathcal{T}_0^{d,t}\}$.
else
  ▷ *the participant $\mathbf{P}_I$ who exists max gain* ◁
  $P_K \xrightarrow{\text{Request}} \mathbf{P}_I$.
  $\mathbf{P}_I \xrightarrow{\{\mathbf{I}_L^{m,j,d,t}, \mathcal{T}_0^{d,t}\}} P_K$.
  P_K generates model.
▷ *Leaf Nodes Building* ◁
P_K generates the leaf weight.

security of threshold, $\mathbf{P}_I$ constructs the encrypted threshold vector $\mathcal{T}_0^{d,1}$ by attribute obfuscation as

$$\mathcal{T}_0^{d,1} = \{[\![\nu^{d,1}]\!], [\![r_1^{d,1}]\!], [\![r_2^{d,1}]\!], \cdots, [\![r_M^{d,1}]\!]\},$$

where $\mathcal{T}_{0,\nu^{d,1}}^{d,1} = [\![r_{\nu^{d,1}}^{d,1}]\!] = [\![\tau^{d,1}]\!]$, $\{r_{i\neq v^{d,1}}\}_{i=1}^M$ are some random numbers, and $\nu^{d,1} \in [1, M]$ is the serial number of the attribute. In this way, the attribute can be hidden in the vector. Then, $\mathbf{P}_I$ sends $\{\mathbf{I}_L^{m,j,d,1}, \mathcal{T}_0^{d,1}\}$ to P_K.

2) ***Leaf Nodes Building***

When a leaf node is built, P_K directly generates the leaf node's weight as Eq. (4) based on the sample index set I of the leaf node and the current gradients.

5.3 DP-based Node Split Algorithm (DPNS)

The DPNS algorithm utilizes differential privacy to privately achieve the tree node generation. Especially, leaf nodes will be built in the same way as the HENS algorithm. Meanwhile, the internal nodes are built as follows.

Step 1: Pre-preparation

First, P_K generates the predicted results and calculates plaintext gradients by Eq. (5). Then, P_K calculates the sensitivity of g^t (resp. h^t) as follows:

$$\begin{aligned}\Delta_{g^t} &= max \parallel g^{\{w^t\}} - g^{\{w'^t\}} \parallel \\ &= 2C, \forall g^t \leq C\end{aligned}$$

Then, it executes differential privacy algorithm to protect gradients as Algorithm 3, where $\sigma \geq \sqrt{2\ln(\frac{1.25}{\delta})}/\epsilon$, and $\mathcal{N} \sim N(\Delta_{g^t}^2\sigma^2)$ is a Gaussian noise generation. Moreover, P_K stores g^t and h^t without noise in the gradient cache to maintain high accuracy when generating leaf weight, and sends $\tilde{g}^t, \tilde{h}^t$ to $\{P_k\}_1^{K-1}$. To construct the first node of the t-th tree, P_K sends $I^{d=1,t}$ to $\{P_k\}_1^{K-1}$.

Step 2: Best split finding

Algorithm 3 Centralized differential privacy algorithm

Input: g^t, ϵ and δ.
Output: $\tilde{g}^t$.
for $i = 1$ to N **do**
  $\tilde{g}_i^t \leftarrow g_i^t / max(1, \frac{\|g_i^t\|}{C})$;
  $\tilde{g}_i^t \leftarrow g_i^t + \mathcal{N} \sim N(\Delta_{g^t}^2\sigma^2)$
g^t will be stored locally and $\tilde{g}^t$ will be sent to P_I.

On receiving $I^{d,t}$, $\{P_k\}_1^K$ update its bucket $\mathcal{B}$ and calculate the local maximum gain $G_I^{k,d,t}$ as Eq. (3). Then, the passive parties $\{P_k\}_{k=1}^{K-1}$ send their maximum gains to P_K. Next, P_K computes

$$G_I^{d,t} = \max\{G_I^{k,d,t} | 1 \leq k \leq K\}.$$

If $G_I^{d,t}$ is from $\{P_k\}_1^{K-1}$, P_K sends $G_I^{d,t}$ to $\mathbf{P}_I$ who owns $G_I^{d,1}$. Then, $\mathbf{P}_I$ seeks m, j of $G_I^{d,1}$ and finishs node construction with P_K.

Step 3: Node construction based on attribute obfuscation

The step is skipped because it is the similar as HESN.

6 Our Scheme

In this section, we describe how to train an efficient and privacy-preserving vertical federated learning model, and provide secure inference for users. The scheme includes three phases, including system initialization, secure model training, and secure inference as shown below.

6.1 System Initialization

The system initialization phase includes two processes, i.e., key distribution and data alignment.

1) ***Key Distribution***

TA first generates an HMAC function with a secret key ssk as $H_{ssk}(\cdot)$ and sends $H_{ssk}(\cdot)$ to all participants. Then, it generates a pair of Paillier's keys (P_{KS}, S_{KS}), and publishes P_{KS}. Meanwhile, P_K generates a pair of Paillier's keys (P_{KT}, S_{KT}) and publishes P_{KT}. Then, it chooses (ϵ, δ) to protect gradients of the trained model.

2) ***Data Alignment***

To train a global model in the vertical federated learning framework, we design a secure alignment algorithm, in which $\{P_k\}_1^K$ match their data with the assistance of the CSP. As shown in Algorithm 4, each P_k first hashes their IDs using $H_{ssk}(\cdot)$ and gets a set of hash values, denoted by $\mathcal{H}_k$ for $1 \leq k \leq K$, which can be finished offline. Then, P_k generates the map $\langle \mathcal{H}_k, \mathcal{ID}_k \rangle$ and sends $\mathcal{H}_k$ to CSP. On receiving $\{\mathcal{H}_k\}_1^K$, CSP chooses the $\mathcal{H}_i$ with the shortest size and generates an N_i-dimensional zero vector $\mathbb{L} = [0, 0, \cdots, 0]$. Then, for each hash value $val_j \in \mathcal{H}_i$, CSP checks whether all of $\{\mathcal{H}_k\}_{1\leq k\neq i\leq K}$ contain it. If yes, CSP sets $\mathbb{L}[j] = 1$. Finally, CSP returns a set of hash values $\{val_j \in \mathcal{H}_i | \mathbb{L}[j] = 1\}$, which corresponds to the IDs that all participants have in common. On receiving $\{val_j \in \mathcal{H}_i | \mathbb{L}[j] = 1\}$, $\{P_k\}_1^K$ identify the corresponding identifies and attribute data for model training.

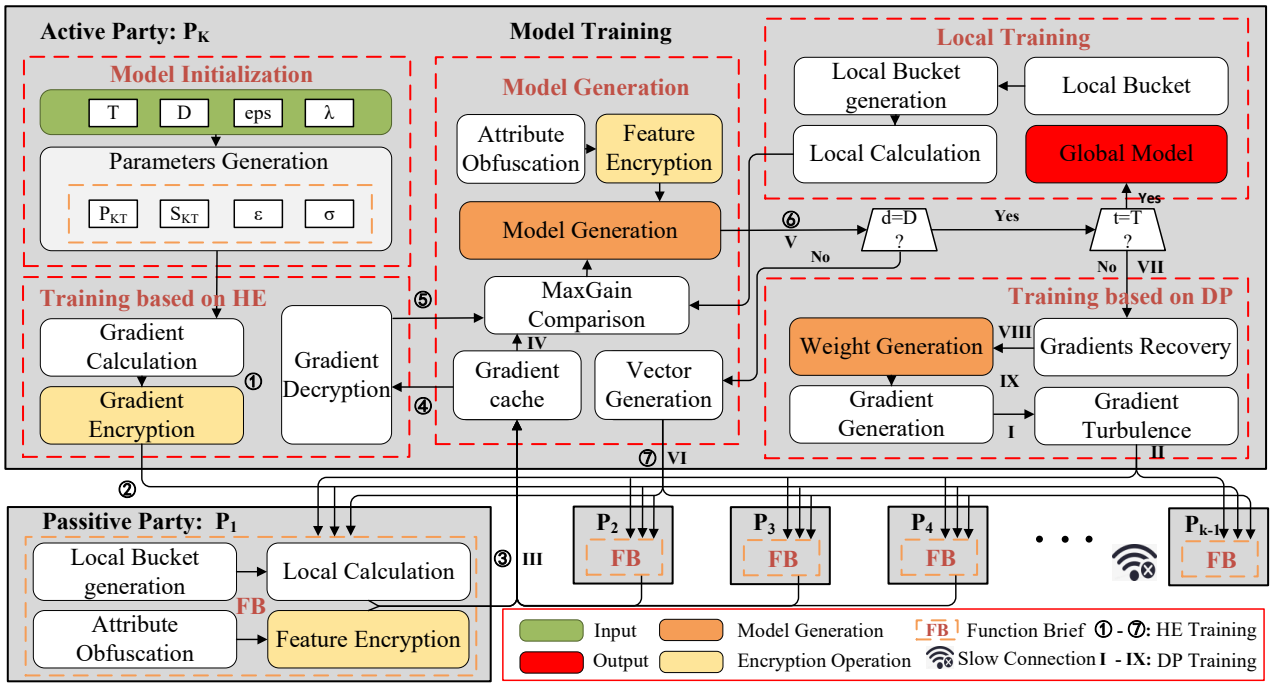

Fig. 3. The overview of model training in ELXGB. The first tree is built by HENS and the other trees are built by DPNS.

Algorithm 4 Secure Data Alignment

$\triangleright$ $\{P_k\}_1^K$ *execute:* $\triangleleft$
$\{P_k\}_1^K \xrightarrow{\mathcal{H}_k}$ to CSP.
$\triangleright$ *CSP executes:* $\triangleleft$
CSP chooses the map $\mathcal{H}_i$ with the shortest size.
CSP generates a zero vector $\mathbb{L} = [0, 0, \cdots, 0]$.
for $val_j \in \mathcal{H}_i$ **do**
  if All of $\{\mathcal{H}_k\}_{1 \le k \ne i \le K}$ contain val_j **then**
    $\mathbb{L}[j] = 1$
CSP $\xrightarrow{\{val_j \in \mathcal{H}_i | \mathbb{L}[j]=1\}} \{P_k\}_1^K$.
$\triangleright$ $\{P_k\}_1^K$ *execute:* $\triangleleft$
$\{P_k\}_1^K$ identify matched data for the next training.

6.2 Secure Model Training

The model training phase includes two processes, i.e., model parameters initialization and tree building. The complete procedure of model training is shown in Fig. 3.

1) *Model Parameters Initialization*

First, P_K sets the maximum tree number to be T, the maximum tree depth to be D, and the initial model weight $\mathbf{w} = (w_1^0, w_2^0, \cdots, w_N^0)$ to be $\mathbf{w} = (0, 0, \cdots, 0)$. Then, it sends T and D to all passive participants.

2) *Tree Building*

The tree model is efficiently built baesd on HENS and DPNS. To ensure the security and performance of model, we give the optimal configuration due to Theorem 1, where all participants collaboratively build the trees $\{f_t(\cdot)\}_{t=1}^T$ as the following steps.

Step 1: $\{P_k\}_1^K$ build the first tree $f_1(\cdot)$ by repeatedly calling the HE-based node split algorithm in Section 5.2 to build the tree node of $f_1(\cdot)$.

Step 2: When training t-th tree, P_K chooses the appropriate parameters ϵ and δ to protect the gradients. Then, $\{P_k\}_1^K$ build the t-th tree $f_t(\cdot)$ by repeatedly calling the DP-based node split algorithm in Section 5.3 to build the tree node of $f_t(\cdot)$ for $2 \le t \le T$.

Finally, P_K obtains the training model $y = \sum_{t=1}^T f_t(\cdot)$.

Correctness of model training. We show the correctness of our model training scheme by proving that the HENS/DPNS algorithms make tree nodes split lossless.

Theorem 1. *The HENS algorithm can make node split lossless and the weight is concentrated on the first tree.*

Proof. We choose the best split by homomorphic encryption on the first tree. The $[\![g_i^t]\!]$ and $[\![h_i^t]\!]$ are encrypted by using Paillier cryptosystem. We can decrypt $[\![\sum_{I_L} g_i]\!]^{m,j,d,t}$ and get lossless $\sum_{I_L} g_i^{m,j,d,t}$ as follows.

$$[\![\sum_{I_L} g_i]\!]^{m,j,d,t} = [\![g_1^t]\!]^{I_1^{m,j,d,t}} \cdot [\![g_2^t]\!]^{I_2^{m,j,d,t}} \cdot \cdots \cdot [\![g_N^t]\!]^{I_N^{m,j,d,t}}$$
$$= [\![g_1^t \cdot I_1^{m,j,d,t} + \cdots + g_N^t \cdot I_N^{m,j,d,t}]\!], t = 1$$

It means that we can calculate the maximum gain G_I as primitive XGBoost, which ensures our model is lossless on the first tree. As Eq. (5), g_i^t is usually calculated by $y_{i,pred}^t - y_i$ and we can find g_i^t is max on the first tree owning to $y_{i,pred}^t$ will be closer to y_i after training. It is obvious that the weight is max on the first tree by Eq. (4). The weight might affect the security and accuracy of model [6], meaning that the first tree should be built by HENS.

Algorithm 5 Direction Obfuscation

Input: $\mathcal{T}_0^{d,t}$ on the current node, $\mathcal{T}_1$
Output: the split direction

▷ *The active party executes:* ◁
P_K generates $\mathcal{T}_3^{d,t}$ and $\mathcal{T}_{3,0}^{d,t} = \mathcal{T}_{0,0}^{d,t}$
P_K selects a random sign $s^{d,t}$ that is 1 or -1.
for $i = 1$ to M **do**
  P_K calculates $c_i^{d,t} = \mathcal{T}_{0,i}^{d,t} \cdot \mathcal{T}_{1,i}$.
  P_K selects two random numbers $n_{1,i}^{d,t}$ and $n_{2,i}^{d,t}$.
  P_K calculates $\hat{c}_i^{d,t} = [\![n_{2,i}^{d,t}]\!]$.
  $\mathcal{T}_{3,i}^{d,t} = ((c_i^{d,t})^{n_{1,i}^{d,t}} \cdot \hat{c}_i^{d,t})^{s^{d,t}}$;
P_K sends $\mathcal{T}_3^{d,t}$ to CSP.
▷ *CSP executes:* ◁
CSP decrypts $\mathcal{T}_{3,0}^{d,t}$ and $\mathcal{T}_{3,\nu^{d,t}}^{d,t}$ by S_{KS}.
if $\mathsf{Dec}(\mathcal{T}_{3,\nu^{d,t}}^{d,t}) < \mathtt{n}/2$ **then**
  CSP sends the split direction $left$ to P_K;
else
  CSP sends the split direction $right$ to P_K;
▷ *The active party executes:* ◁
if $s^{d,t} = -1$ **then**
  P_K changes the direction.

Theorem 2. *The DPNS algorithm can also make node split almost lossless.*

Proof. The $\tilde{g}_i^t$ and $\tilde{h}_i^t$ are added Gaussian Noise on the other trees that protect data labels. When the training data is large, $<\tilde{g}^t>^2$ approximates to $<g^t>^2$. Therefore, gains in each I can be calculated by Eq. (3). However, to maintain the model accuracy, P_K utilizes g^t and h^t from the gradient cache to calculate w^t on the current tree by Eq. (4), which are secret to P_I. Moreover, the corresponding experiments shows that our model is highly accurate as the XGBoost.

6.3 Secure Inference

In this section, we describe how to finish the secure inference for users of P_K. The secure inference includes three phases: attribute obfuscation, direction obfuscation, and service response.

During the inference, to provide a personal service for the users, P_K finishes the seucre inference with the assistance of CSP. The identified CSP needs to ask a request for S_{KT} to TA, and TA sends S_{KT} securely to CSP.

1) ***Attribute Obfuscation***

To obtain a personal service from P_K, the registerd user gets the serial numbers about attributes and P_{KS} from TA. According to the serial numbers, the user utilizes the local information to generate the encrypted vector $\mathcal{T}_1$ by P_{KS} as follows

$$\mathcal{T}_1 = \{0, [\![-u_1]\!], [\![-u_2]\!], \cdots, [\![-u_M]\!]\} \tag{6}$$

where $u_{i:1\leq i\leq M}$ is the specific information corresponding to each attribute of the user, and i is the serial number corresponding to the attribute. The encrypted vector $\mathcal{T}_1$ is sent to the active party P_K, which can avoid knowing which attribute is used to split.

2) ***Direction Obfuscation***

On receiving $\mathcal{T}_1$, P_K executes the direction obfuscation as Algorithm 5. First, P_K initializes the encrypted vector $\mathcal{T}_3^{d,t}$ for the d-th node of the t-th tree. Then,P_K chooses a random sign $s^{d,t}$, and numbers $\{n_{1,i}^{d,t}\}_{i=1}^M$, $\{n_{2,i}^{d,t}\}_{i=1}^M$, satisfying $s^{d,t} = (-1|1), \{n_{1,i}^{d,t}\}_{i=1}^M \gg \{n_{2,i}^{d,t}\}_{i=1}^M > 0$, which are used to prevent the split direction from being known by the CSP. Satisfying $\mathcal{T}_{3,0}^{d,t} = \mathcal{T}_{0,0}^{d,t}$, P_K calculates $\{\mathcal{T}_{3,i}^{d,t}\}_{i=1}^M$ as follows,

$$\{\mathcal{T}_{3,i}^{d,t}\}_{i=1}^M = ((\mathcal{T}_{0,i}^{d,t}\mathcal{T}_{1,i})^{n_{1,i}^{d,t}}\hat{c}_i^{d,t})^{s^{d,t}}$$

where $\hat{c}_i^{d,t} = [\![n_{2,i}^{d,t}]\!]$, and sends $\mathcal{T}_3^{d,t}$ to CSP.

On receiving the encrypted vector, CSP decrypts the first value of the vector by S_{KS} and gets the attribute number $\nu^{d,t}$. Then, CSP decrypts $\mathcal{T}_{3,\nu^{d,t}}^{d,t}$ and sends the direction to P_K. If $s^{d,t} = -1$, P_K needs to change the direction. After getting the true split direction of the node, P_K moves to the next node by it. P_K and CSP will repeat the above steps until reaching the leaf node. Because of the characteristic of parallel trees, P_K makes at most D times of communications with CSP to reach the leaf nodes of each tree.

3) ***Service Response***

When reaching the leaf nodes of each tree, P_K calculates $\sum w$ from these leaf nodes. Then, P_K calculates the prediction by Eq. (5). Utilizing the prediction, P_K can provide a personal and accurate service for the user.

Correctness of model inference. We show the correctness of the model inference process by proving that the direction obfuscation process can provide secure and accuracy inference for users.

Theorem 3. *Direction obfuscation can provide secure and accuracy inference for users.*

Proof. It can be proved that direction obfuscation provides secure and accuracy inference for users as follows:

$$\begin{aligned}\mathsf{Dec}(\mathcal{T}_{3,\nu^{d,t}}^{d,t}) &= \mathsf{Dec}((([\![\tau^{d,t}]\!] \cdot [\![-u_{\nu^{d,t}}]\!])^{n_{1,\nu^{d,t}}^{d,t}} \cdot [\![n_{2,\nu^{d,t}}^{d,t}]\!])^{s^{d,t}}) \\ &= \mathsf{Dec}([\![s^{d,t}(n_{1,\nu^{d,t}}^{d,t}(\tau^{d,t} - u_{\nu^{d,t}}^{d,t}) + n_{2,\nu^{d,t}}^{d,t})]\!]) \\ &= s^{d,t}(n_{1,\nu^{d,t}}^{d,t}(\tau^{d,t} - u_{\nu^{d,t}}^{d,t}) + n_{2,\nu^{d,t}}^{d,t})\end{aligned}$$

Therefore, $s^{d,t}(n_{1,\nu^{d,t}}^{d,t}(\tau^{d,t} - u_{\nu^{d,t}}^{d,t}) + n_{2,\nu^{d,t}}^{d,t}) > \mathtt{n}/2$ represents right and $s^{d,t}(n_{1,\nu^{d,t}}^{d,t}(\tau^{d,t} - u_{\nu^{d,t}}^{d,t}) + n_{2,\nu^{d,t}}^{d,t}) < \mathtt{n}/2$ represents left. CSP sends the split direction to P_K. When $s^{d,t}$ is -1, P_K needs to reverse the direction. Otherwise, the direction is remained, which represents how to split to get the weight.

7 Security Analysis

In this section, we analyze the security of the proposed scheme. Since ELXGB contains three phases, we sequentially analyze and prove their security through the simulation-based real/ideal worlds model. The real world model works as our ELXGB scheme, and the ideal world is simulated based on our scheme. If the views of adversaries in real and ideal worlds are indistinguishable, our scheme is secure. Specifically, the real and ideal worlds can be defined as follows.

Real World: In the real world, ELXGB contains three phases, including data alignment, model training, and inference service as follows.

- **Phase 1 Data Alignment.** $\{P_k\}_1^K$ hash their IDs using $H_{ssk}(\cdot)$ and get $\mathcal{H}_k$. Then, $\{P_k\}_1^K$ send $\mathcal{H}_k$ to

CSP. After matching the aligned data, CSP sends a set of hash values $\{val_j \in \mathcal{H}_i | \mathbb{L}[j] = 1\}$ to $\{P_k\}_1^K$.

- **Phase 2.1 Model Training during the First Tree.** To find the best split of the first tree, P_K sends the encrypted gradients $[\![g^1_{i:1\le i\le N}]\!]$ and $[\![h^1_{i:1\le i\le N}]\!]$ to $\{P_k\}_1^{K-1}$. To securely build the node, $\mathbf{P}_I$ constructs the encrypted threshold vector $\mathcal{T}_0^{d,1}$ by attribute obfuscation and sends the encrypted node $\{\mathbf{I}_L^{m,j,d,1}, \mathcal{T}_0^{d,1}\}$ to P_K.
- **Phase 2.2 Model Training during the Other Trees.** To find the best split of the other trees, P_K sends the noise gradients $\tilde{g}^t, \tilde{h}^t$ to $\{P_k\}_1^{K-1}$. To securely build the node, $\mathbf{P}_I$ constructs the encrypted threshold vector $\mathcal{T}_0^{d,t}$ by attribute obfuscation and sends the encrypted node $\{\mathbf{I}_L^{m,j,d,t}, \mathcal{T}_0^{d,t}\}$ to P_K.
- **Phase 3 Inference Service.** To get a personal service, the user sends the encrypted vector $\mathcal{T}_1$ to P_K. On receiving $\mathcal{T}_1$, P_K calculates $\{\mathcal{T}_{3,i}^{d,t}\}_{i=1}^M$ and sends $\mathcal{T}_3^{d,t}$ to CSP. CSP decrypts $\mathcal{T}_{3,\nu^{d,t}}^{d,t}$ and sends the direction to P_K.

Ideal World: In the ideal world, there exist some PPT adversaries and simulators in each phase. The simulators need to simulate the real world for adversaries as follows.

- **Phase 1 Data Alignment.** The $\mathcal{S}_1$ chooses some random and fixed-length string $\{\mathcal{H}_k\}_{sim}$. Then, $\mathcal{S}_1$ sends $\{\mathcal{H}_k\}_{sim}$ to $\mathcal{A}_1$. After matching the aligned data, $\mathcal{S}_2$ sends a random set $\{\{\mathbb{L}|\mathbb{L}[i] = 1/0\}\}_{sim}$ to $\mathcal{A}_2$.
- **Phase 2.1 Model Training during the First Tree.** First, $\mathcal{S}_1$ chooses random gradients, encrypts, and gets $[\![\{g^1_{i:1\le i\le N}\}_{sim}]\!]$ and $[\![\{h^1_{i:1\le i\le N}\}_{sim}]\!]$ to $\mathcal{A}_1$. Then, $\mathcal{S}_2$ constructs the encrypted random threshold vector and bit vector $\{\mathcal{T}_0^{d,1}\}_{sim}, \{\mathbf{I}_L^{m,j,d,1}\}_{sim}$, and sends them to $\mathcal{A}_2$.
- **Phase 2.2 Model Training during the Other Trees.** First, $\mathcal{S}_1$ sends the random gradients $\{\tilde{g}^t\}_{sim}, \{\tilde{h}^t\}_{sim}$ to $\mathcal{A}_1$. Then, $\mathcal{S}_2$ constructs the encrypted random threshold vector and bit vector $\{\mathcal{T}_0^{d,t}\}_{sim}, \{\mathbf{I}_L^{m,j,d,t}\}_{sim}$, and sends them to $\mathcal{A}_2$.
- **Phase 3 Inference Service.** $\mathcal{S}_1$ sends the encrypted random vector $\{\mathcal{T}_1\}_{sim}$ to $\mathcal{A}_1$. Then, $\mathcal{S}_2$ encrypts a random vector $\{\mathcal{T}_3^{d,t}\}_{sim}$ and sends it to $\mathcal{A}_2$.

Definition 1 ***Security of Data Alignment.*** Data alignment is selectively secure iff for any two PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, there exist two simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ to simulate an ideal world, where $\mathcal{A}_1$ and $\mathcal{A}_2$ just distinguish the real and ideal worlds with a negligible probability.

Theorem 4. *Data alignment is secure where CSP cannot get data order and private ID of participants and $\{P_k\}_1^K$ can not obtain any useful information except the aligned data during the data alignment.*

During data alignment, IDs are hashed by $H_{ssk}(\cdot)$, shuffled, and sent to CSP. Owning to shuffled indexes and the security of ssk, they are random[31], meaning that $\mathcal{A}_1$ cannot distinguish $\mathcal{H}_k$ and $\{\mathcal{H}_k\}_{sim}$. Moreover, because $\{val_j \in \mathcal{H}_i | \mathbb{L}[j] = 1\}$ and $\{\{\mathbb{L}|\mathbb{L}[i] = 1/0\}\}_{sim}$ are both random, $\mathcal{A}_2$ cannot distinguish the set from the real or ideal worlds. Therefore, the data alignment is secure.

Definition 2 ***Security of Model Training on the First Tree.*** Model training on the first tree is selectively secure iff for any two PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, there exist two simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ to simulate an ideal world, where $\mathcal{A}_1$ and $\mathcal{A}_2$ just distinguish the real and ideal worlds with a negligible probability.

Theorem 5. *Model training on the first tree is secure where $\{P_k\}_1^{K-1}$ cannot obtain any gradient or weight information, and P_K cannot obtain any node information.*

As we know,P_K owns the private key of gradient information encryption, and CSP owns the private key of node information encryption. When training the first tree, the gradients are encrypted and sent to $\{P_k\}_1^{K-1}$. Because Paillier is the IND-CPA [32], $\mathcal{A}_1$ cannot distinguish the gradients from the real or ideal worlds (resp. weight information). Therefore, $\{P_k\}_1^{K-1}$ cannot obtain any gradient or weight information. Moreover, the threshold information is encrypted by the attribute obfuscation and sent to P_K, meaning P_K cannot distinguish the threshold from the real or ideal worlds. All node attributes from $\{P_k\}_1^{K-1}$ are hidden, and the bit information is limited by the node numbers, meaning that $\mathcal{A}_2$ does not know which attribute is related to $\mathbf{I}$. Therefore, P_K cannot obtain any node information. So, the model training on the first tree is secure.

Definition 3 ***Security of Model Training on the Other Trees.*** Model training on the other trees is selectively secure iff for any two PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, there exist two simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ to simulate an ideal world, where $\mathcal{A}_1$ and $\mathcal{A}_2$ just distinguish the real and ideal worlds with a negligible probability.

Theorem 6. *Model training on the other trees is secure where $\{P_k\}_1^{K-1}$ cannot obtain any gradient or weight information, and P_K cannot obtain any node information.*

We analyze two different learning tasks that involve binary classification and regression tasks. It is clear that the noise gradients $\tilde{g}_i^t$ ($g_i^t = \hat{y}_i^t - y_i^t$) make $\{P_k\}_1^{K-1}$ confused about data labels on regression task. However, it may be easy to guess data labels because there are only two results (0 or 1) on a binary classification task. When the gradient $g_i > 0$ or < 0, we can think the true label y_i is 0 or 1, respectively. Therefore, it is insecure that P_K transfers the primitive gradients to the passive parties. In our scheme, P_K selects the secure parameters (ϵ, δ) and adds noise to gradients to meet the following condition:

$$Pr(g_i^t > 0 \ \& \ y_i^t = 0) + Pr(g_i^t < 0 \ \& \ y_i^t = 1) \approx 50\%. \quad (7)$$

When the probability is close to 50%, A_1 cannot distinguish which label (1/0) corresponds to the noise gradient. Therefore, $\{P_k\}_1^{K-1}$ cannot obtain any gradient information (resp. weight). The security of node information has been proven in Theorem 5. So, model training on the other trees is secure.

Definition 4 ***Security of Model Inference.*** Model inference is selectively secure iff for any two PPT adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$, there exist two simulators $\mathcal{S}_1$ and $\mathcal{S}_2$ to simulate an ideal world, where $\mathcal{A}_1$ and $\mathcal{A}_2$ just distinguish the real and ideal worlds with a negligible probability.

Theorem 7. *Model inference is secure where CSP and P_K cannot obtain the attribute information of users and split threshold of $\{P_k\}_1^{K-1}$.*

During the inference, all information is encrypted by the private key of CSP. During the inference, the information of

TABLE 2
Hyperparameters of ELXGB model

Hyperparameter	Range	Default
The number of participants n	/	4
Size of ssk	/	512
Paillier's key size for training	/	512
Paillier's key size for inference	/	512
Privacy parameter ϵ	[2,4,6,8,10]	10
Maximum tree number T	[2,3,4,5,6]	5
Maximum depth D	[2,3,4,5,6]	3
Learning Rate	/	0.3
Regularization item λ	/	1
The numbers of I in a Bucket $\frac{1}{eps}$	/	32

users is encrypted and sent to P_K. Because Paillier is the IND-CPA [32], $\mathcal{A}_1$ cannot distinguish the user information from the real or ideal worlds.

When comparing one node, P_K selects two small random numbers $n^{d,t}_{1,\nu^{d,t}}$, $n^{d,t}_{2,\nu^{d,t}}$ and a random sign $s^{d,t}$ to protect the difference between $\tau^{d,t}$ and $u^{d,t}_{\nu^{d,t}}$. It means that CSP just gets messages $s^{d,t}(n^{d,t}_{1,\nu^{d,t}}(\tau^{d,t} - u^{d,t}_{\nu^{d,t}}) + n^{d,t}_{2,\nu^{d,t}})$ obtained through the decryption and regards them as the judgment of directions, in which it cannot get any useful information from $\{P_k\}_1^{K-1}$, users, or the global model. By the algorithms for the approximate common divisor problem [33], CSP can solve the following problem $x_i = p \cdot q_i + r_i$ and get p. To protect the user's information $u_{i:1\leq i\leq M, i\neq\nu^{d,t}}$, the passive parties generate some random numbers $r_i^{d,t}$, which makes CSP only get $s_i^{d,t}(n^{d,t}_{1,i}(r_i^{d,t} - u_i) + n^{d,t}_{2,i})$, where i represents the same attribute. It can guarantee that the p of the above problem is different for each calculation. Therefore, $\mathcal{A}_2$ cannot distinguish $\mathcal{T}_3^{d,t}$ from the real or ideal worlds.

So, the model inference is secure. CSP and P_K cannot obtain attribute information of users and split threshold of $\{P_k\}_1^{K-1}$. It means that the global model is protected.

8 Performance Evaluation

In this section, we evaluate the performance of our scheme over two real-world datasets and compare it with some representative schemes, including SecureBoost [6] and PIVODL [7], from the aspects of model accuracy, computational costs, and communication overheads.

8.1 Experimental Setting

We implement the training and prediction processes of ELXGB with Java 12.0.1. The experiments were conducted on ThinkPad-P53 machine equipped with 23.1 GB RAM and Intel Core i5-9400H 2.50GHz processor, running on ubuntu 18.04.6. Moreover, the used hyperparameters of the ELXGB model are presented in TABLE 2. Two representive datasets are used to evaluate the performance of ELXGB. The training and test datasets occupy 80% and 20% of the entire data samples, respectively.

TABLE 3
Run-time with different numbers of participants

n	4			5		
$\overline{N}$	2^{12}	2^{16}	2^{18}	2^{12}	2^{16}	2^{18}
mPSI (ms)	1460	2910	9320	1620	3100	9490
Ours (ms)	6	103	501	7	124	620

Credit Card [34]: The dataset contains 23 features and 30000 samples, which is a classification task to predict whether a person will make a payment on time.

Bank Marketing [35]: The dataset has 17 features and 45211 instances, which records direct marketing campaigns of a Portuguese banking institution. The classification task can predict whether clients will subscribe for time deposit.

In our performance evaluations, all experiments show average results and each repeated five times independently.

8.2 Model Accuracy

In this subsection, we compare the influence on model performances as the maximum tree numbers, the maximum tree depths, and privacy parameters change from Fig. 4. First, we can see that the average accuracy by the global proposal is about 83% and 90% for Credit card and Bank marketing datasets, respectively. Second, compared with the primitive XGBoost, we find that ELXGB almost achieves the same accuracy. Since we handle the weight of leaf nodes by restoring the true gradient on P_K, it can reduce the impact of noise on the model accuracy. Moreover, it is normal that the model accuracy has a slight fluctuation because of the global proposal to build the model. Third, we can find that, as ϵ increases, the variation in the accuracy of our model is slight. Therefore, we can choose the more secure parameters to train an accurate model.

8.3 Computational Costs

In this subsection, we analyze the running time on data alignment and compare it with recent scheme [36]. Then, we show the time cost during training and service, analyze the influence of the tree number and the tree depth on the model and compare our scheme with XGBoost, SecureBoost and PIVODL.

1) Data Alignment: We choose SHA512 as the hash algorithm and finish our experiments with different intersection sizes over different participants. From TABLE 3, with the increase of training data, the time cost grows faster. However, in general, the cost of our algorithm is small. Compared with Circuit/Quorum scheme [36], where the statistical security parameter is 40 and the computational security parameter is 128, when the number of participants is 4 and the average total data is 2^{16}, its time costs about 2910ms and our time just costs about 103ms. Therefore, the secure data alignment can meet our security needs to finish data alignment efficiently.

2) Model Training: In order to obtain a better contrast effect, we compare the training time of our scheme with that of XGBoost, SecureBoost, and PIVODL. The length of

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.2024.3394706

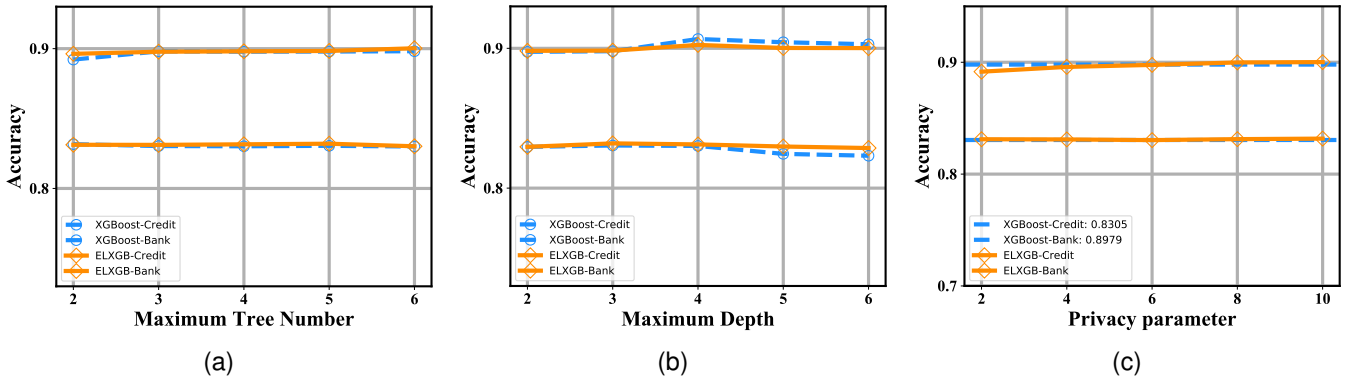

Fig. 4. The accuracy over different aspects on different datasets. (a). Accuracy with different T; (b). Accuracy with different D; (c). Accuracy with different ϵ;

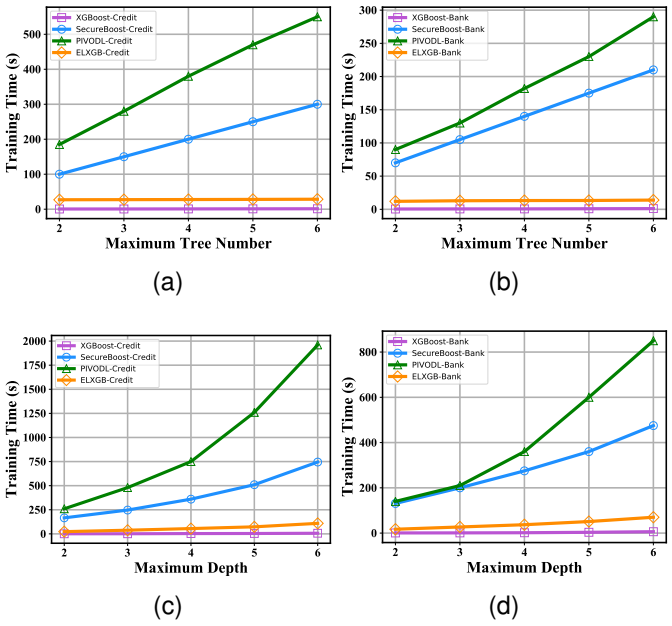

Fig. 5. The training time over different aspects on different datasets. (a)-(b). The training time with different T; (c)-(d). The training time with different D;

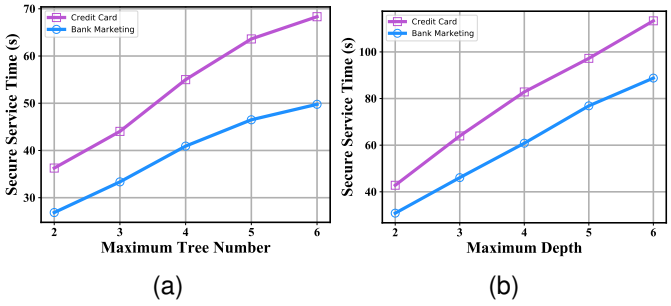

Fig. 6. The service time over different aspects on different datasets. (a). The service time with different T; (b). The service time with different D.

Paillier key in all secure schemes is 512. From Fig. 5 (a)-(b), we can know that the training time cost in our scheme grows slowly with the tree numbers increasing. Unlike other schemes, our scheme just uses the encryption cryptosystem in the first tree. Therefore, our cost concentrates on the first tree, and we reduce overhead in later training. By contrast, the training time in other schemes grows linearly with the maximum tree number increasing. From Fig. 5 (c)-(d), we can find that the time overhead increases faster as the maximum tree depth grows than as the maximum tree number grows. Moreover, with the maximum tree depth increasing, the training time of our scheme grows slower than that of other schemes. We can find that the time has increased alarmingly when utilizing the encryption cryptosystem in XGBoost. For example, When the tree number is 5 and the tree depth is 3 on Credit Card, the running time in primitive XGBoost costs about 1s. However, PIVODL costs about 480s, and SecureBoost costs about 209s. With the tree structure changing, their computation cost has drastic changes. However, the expensive encryption overhead as Eq. (5) can be finished offline in our scheme. Therefore, our scheme can efficiently train the global model independently of the model structure.

3) Secure Inference: On different datasets, we all choose 1000 samples to test the secure service process, and the size of service key is 512. From Fig. 6 (a)-(b), we can know that the time of secure service increases slower with maximum tree number growing than with maximum tree depth growing. Because of the characteristics of parallel tree, we can simultaneously predict the weight. Therefore, as the maximum tree number increases, the time cost grows slower. It is clear that the cost of service has a strong relationship to the model structure. Moreover, we can find that the service time cost on different datasets is also different. The time cost on Credit Card is higher than on Bank Marketing because the former has more dimensions than the latter.

8.4 Communication Cost

In this subsection, we evaluate the total communication overhead of the training process affected by the maximum tree number and maximum depth. The participants are 4, including an active party and three passive parties.

From Fig. 7 (a)-(b), we can find that as the maximum tree number increases, the overhead in our scheme keeps stable, and PIVODL's and SecureBoost's overhead grows faster.

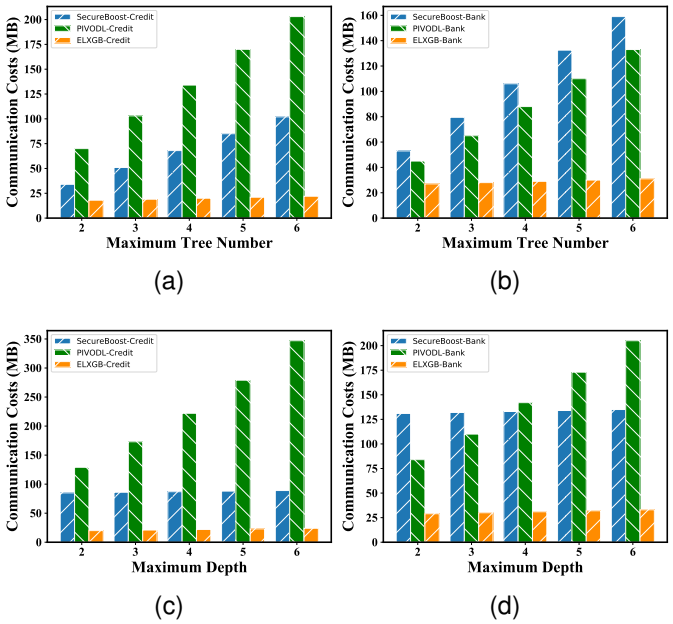

Fig. 7. The communication cost over different aspects on different datasets. (a)-(b). The communication cost with different T; (c)-(d). The communication cost with different D.

From Fig. 7 (c)-(d), we can know our scheme and SecureBoost are hardly unaffected by the increase of tree depth. However, SecureBoost's communication cost is about five times ours, and the communication overhead in PIVODL grows faster as the maximum depth changes. We can also find that our scheme's communication cost is more higher on Credit Card than Bank Marketing because the former has more training data than the latter. Therefore, the communication cost of our scheme is affected by the number of training data.

In our scheme, most overhead is concentrated on the first tree built. The encrypted value of gradient and hessian by HE will be transmitted to all passive parties. However, the tree structure does not affect the communication overhead. Therefore, when the tree structure is richer, our scheme has more advantages than other schemes.

9 Related Work

Based on the distribution characteristics of the data, federated learning can be classified into horizontal FL, vertical FL, and federated transfer learning [3]. In vertical FL, participants' data sets share the same sample ID space but differ in attribute space. According to the data distribution, privacy-preserving machine learning algorithms have been proposed, which solve regression, classification, and more. For example, Hardy *et al.* [37] proposed a vertical federated learning scheme to train a privacy-preserving logistic regression model, which applies Taylor approximation to the loss and gradient functions, and homomorphic encryption is adopted for privacy-preserving computations. Wang *et al.* [28] proposed a privacy-preserving vertical FL for native bayesian classification. Based on Paillier cryptosystem and random masking techniques, they aggregate an efficient and privacy-preservation prediction model over vertically distributed data. Zhao *et al.* [38] proposed a data aggregation matrix construction algorithm, which aggregates the vertically partitioned data for logistic regression training. The model can be securely trained without multi-round interactions between the third party and participants. Wu *et al.* [39] proposed a privacy-preserving solution based on HE and MPC for vertical tree-based models, which guarantees that no intermediate information is disclosed during the execution. Tian *et al.* [40] propose a framework named FederBoost to support running GBDT over vertical and horizontal data. Based on ϵ-LDP, it can securely publish bucket information to complete gradient aggregation for the vertical data. Zheng *et al.* [41] propose the first system framework named Privet to support training and predicting for the gradient boosted decision tables on vertical data. Based on lightweight cryptography, it can well protect data information dispersed among various participants.

In our work, we propose an efficient and privacy-preserving vertical FL for XGBoost training on the premise of meeting security requirements, which are mentioned in section 2. We discuss and compare existing works with it, which can be mainly classified into HE-based schemes and MPC-based schemes, according to privacy-preservation techniques. TABLE 4 summarizes the major related works in terms of privacy preservation and performance adopted.

HE-based schemes. Cheng *et al.* [6] proposed a lossless federated learning framework called SecureBoost, which sums gradients by Paillier on the passive parties and decrypts them on the active party. They maintain high model accuracy. However, massive communication and calculation costs are unbearable during each training epoch. Moreover, weight sent to the corresponding passive party leaks data labels. Zhu *et al.* [7] proposed a novel vertical federated learning scene where data labels are distributed among multiple clients. Therefore, the scheme can be treated as the variant of horizontal federal learning scheme. They locally calculate the partial sum of gradients, encrypt them by HE, and send them to the aggregation server for the whole sum. To protect data labels, they add noise to weight by DP. However, data labels might be recovered by model inversion attacks [10], [19]. Moreover, with the increase of training epochs, their training scheme will spend unavoidable communication overhead. Chen *et al.* [8] proposed SecureBoost+ that is both novel and improved from the prior work SecureBoost. Compared with SecureBoost, SecureBoost+ utilizes data packing to decrease the encryption overhead of the active side by half. Moreover, by the ciphertext compressing, SecureBoost+ further reduces the communication overhead between the passive parties and the active party and reduces the decryption times for the active party. The optimized approaches can be used in the HENS to further improve the efficiency of ELXGB.

MPC-based schemes. Le *et al.* [11] proposed a privacy-preserving XGBoost scheme called FedXGBoost-SSM for federated learning. FedXGBoost-SSM utilizes characteristics of the matrix to protect data labels and data distributions, while existing some risk of leakage to data distributions. Moreover, they also give another approach called FedXGBoost-LDP to train the model, which has more efficiency for training. However, it is just suitable for regression tasks, and model accuracy decreases. Fang *et al.* [12]

TABLE 4
Comparisons of Security and Performance in Vertical Federated XGBoost Schemes

Schemes	Privacy Preservation				Accuracy	The Global Model	Efficiency
	Data	Bucket	Threshold	Label			
SecureBoost [6]	✓	✓	✗	✗	High	Distributed	Low
PIVODL [7]	✓	✓	✗	✗	Less High	Distributed	Less High
FedXGBoost-SSM [11]	✓	✗	✗	✓	High	Distributed	Less High
SS-XGB [12]	✓	✓	✗	✓	Less High	Distributed	Low
MP-FedXGB [13]	✓	✓	✗	✓	High	Distributed	Less High
SGBoost [16]	✓	✓	✗	✓	High	Centralized	Less High
ELXGB	✓	✓	✓	✓	***High***	***Centralized***	***High***

proposed to build large-scale secure XGBoost by leveraging hybrid secure multi-party computation techniques. They provide two approaches to improve the model performance: HEP-XGB and CRP-XGB, respectively. Both approaches can protect data labels and data distributions well. However, their efficiency is unacceptable, and the model accuracy declines due to the reconstruction of the sigmoid function. Xie $et\ al.$ [13] proposed a secret sharing scheme with lower overhead. Their framework can provide secure training and prediction to protect data labels and data order well. However, these processes need all participants to keep continuously online, which is vital to finish the accurate inference. Zhao $et\ al.$ [16] proposed an efficient and privacy-preserving vertical federated boosting tree framework based on secure two-party computation. The passive parties divide the bucket into two parts through secret sharing, which are respectively sent to the active party and CSP. Therefore, the training and inference processes can be finished flexibly between the two sides. The SHE [42] as the vital technique of the scheme has more efficient performance compared with other HE schemes. However, the newest research states that SHE can be attacked by the approximate common divisor problem [33], meaning the scheme might leak sensitive information. Moreover, the global model is leaked to CSP, which increases the risk of sensitive data leakage [33] [19]. In addition, the inference process needs to traverse every node of the tree model, which generates huge overhead.

10 Discussion

In this section, we discuss here security and optimization extensions that can be integrated to ELXGB.

1) Why is HENS utilized to construct the first tree and DPNS utilized to construct the other trees? In XGBoost, we find that its model weight is concentrated on the first tree. It means that the weight might leak some extra information, such as data labels [6]. Therefore, it deserves higher security guarantees to protect weight. Moreover, we further prove our theorem by experiments as TABLE 5. When utilizing DPNS to construct the model, the curious passive parties might utilize model inversion attacks to guess the true labels. For example, after receiving the noise gradients, they can calculate the probability of Eq. (7). If they own some true training labels, they can successfully verify the possibility of these guesses. And it is insecure when the possibility is very high or very low, e.g., if one adversary knows the possibility is 20%, it will have 80% possibility to guess the true labels on the contrary. In TABLE 5, we can find the guessing possibility on the first tree is higher than on other trees, and it means the data labels might be leaked. Therefore, considering the balance between privacy and performance, we choose HENS to construct the first tree and choose DPNS to construct the other trees.

TABLE 5
Guess possibility over different trees with different ϵ

	$\epsilon = 2$	$\epsilon = 4$	$\epsilon = 6$	$\epsilon = 8$	$\epsilon = 10$
first tree on DP algorithm (prohibition)	60.01%	68.23%	75.60%	82.29%	87.73%
second tree	53.83%	57.18%	59.88%	63.84%	66.52%
third tree	53.58%	55.98%	59.16%	61.65%	63.92%
fourth tree	52.27%	55.44%	59.04%	61.07%	62.87%
fifth tree	52.88%	55.84%	57.73%	60.71%	63.12%

2) Can HENS be further optimized? In our scheme, we give the optimal configuration, where we choose HENS to construct the first tree and choose DPNS to construct the other trees. It is clear that the expensive overhead is concentrated on the first tree. For P_K, it needs to encrypt $2 \cdot N$ data for $\{g_i^1\}_{i=1}^N$ and $\{h_i^1\}_{i=1}^N$. For the i-th passive party, it needs to encrypt $2 \cdot \frac{1}{eps} \cdot M_i \cdot 2^{D-1}$ aggregated gradients, where 2^{D-1} represents the node numbers. To decrease the encryption times, we can use the data-packing and ciphertext compressing to optimize Paillier as SecureBoost+ [8]. It can decrease the encryption overhead of the active side by half and the encryption overhead of the passive party by η, where η represents the compression times. When the data number N is large, the total encryption times can be approximated as N. Therefore, to further decrease it, CKKS [43] is proposed as the encryption scheme, which supports data-packing and is more lightweight. By CKKS, the encryption times can be reduced to several times to publish gradients. Therefore, HENS can be further optimized by data-packing.

11 Conclusion

In this paper, we propose an efficient and privacy-preserving vertical federated boosting tree framework named ELXGB, which can achieve high-accuracy training and inference securely and efficiently. We have carefully designed the processes of training and inference, which can resist model inversion attacks and property inference attacks. Experimental results also demonstrate that ELXGB can train a high-quality model under the premise of ensuring security and efficiency. In the future, we plan to train the model by a trust executed environment [44] to further improve the performance and security of federated XGBoost model training.

Acknowledgements

This work was supported by National Natural Science Foundation of China (U22B2030, 62302360), Shaanxi Provincial Key Research and Development Program (2023-ZDLGY-35), Natural Science Basic Research Plan in Shaanxi Province of China (2023-JC-QN-0699), China Postdoctoral Science Foundation (2022M722498), Qin Chuangyuan Cited High-level Innovative and Entrepreneurial Talents Project (QCYRCXM-2022-244).

References

[1] European Parliament and Council of the European Union. (2016, Apr.) General Data Protection Regulation. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj/

[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 2017, pp. 1273–1282.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

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

[5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. abs/1908.07873, 2019.

[6] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, and Q. Yang, "Secureboost: A lossless federated learning framework," *IEEE Intelligent Systems.*, vol. 36, no. 6, pp. 87–98, 2021.

[7] H. Zhu, R. Wang, Y. Jin, and K. Liang, "PIVODL: privacy-preserving vertical federated learning over distributed labels," *CoRR*, vol. abs/2108.11444, 2021.

[8] W. Chen, G. Ma, T. Fan, Y. Kang, Q. Xu, and Q. Yang, "Secureboost+ : A high performance gradient boosting tree framework for large scale vertical federated learning," *CoRR*, vol. abs/2110.10927, 2021.

[9] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 1592. Springer, 1999, pp. 223–238.

[10] M. Fredrikson, E. Lantz, S. Jha, S. M. Lin, D. Page, and T. Ristenpart, "Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing," in *USENIX Security Symposium*. USENIX Association, 2014, pp. 17–32.

[11] N. K. Le, Y. Liu, Q. M. Nguyen, Q. Liu, F. Liu, Q. Cai, and S. Hirche, "Fedxgboost: Privacy-preserving xgboost for federated learning," *CoRR*, vol. abs/2106.10662, 2021.

[12] W. Fang, D. Zhao, J. Tan, C. Chen, C. Yu, L. Wang, L. Wang, J. Zhou, and B. Zhang, "Large-scale secure XGB for vertical federated learning," in *CIKM*. ACM, 2021, pp. 443–452.

[13] L. Xie, J. Liu, S. Lu, T. Chang, and Q. Shi, "An efficient learning framework for federated xgboost using secret sharing and distributed optimization," *ACM Transactions on Intelligent Systems and Technology (TIST).*, vol. 13, no. 5, pp. 77:1–77:28, 2022.

[14] P. Mohassel and P. Rindal, "Aby^3: A mixed protocol framework for machine learning," in *CCS*. ACM, 2018, pp. 35–52.

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

[16] J. Zhao, H. Zhu, W. Xu, F. Wang, R. Lu, and H. Li, "Sgboost: An efficient and privacy-preserving vertical federated tree boosting framework," *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2022.

[17] A. Beimel, "Secret-sharing schemes: A survey," in *IWCC*, ser. Lecture Notes in Computer Science, vol. 6639. Springer, 2011, pp. 11–46.

[18] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: Definitions and challenges," in *TCC*, ser. Lecture Notes in Computer Science, vol. 6597. Springer, 2011, pp. 253–273.

[19] M. Song, Z. Wang, Z. Zhang, Y. Song, Q. Wang, J. Ren, and H. Qi, "Analyzing user-level privacy attack against federated learning," *IEEE Journal on Selected Areas in Communications.*, vol. 38, no. 10, pp. 2430–2444, 2020.

[20] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *CCS*. ACM, 2016, pp. 308–318.

[21] N. Wang, X. Xiao, Y. Yang, J. Zhao, S. C. Hui, H. Shin, J. Shin, and G. Yu, "Collecting and analyzing multidimensional data with local differential privacy," in *ICDE*. IEEE, 2019, pp. 638–649.

[22] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *CoRR*, vol. abs/2003.02133, 2020.

[23] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography*. ACM, 2019, pp. 307–328.

[24] M. Fang, X. Cao, J. Jia, and N. Z. Gong, "Local model poisoning attacks to byzantine-robust federated learning," in *USENIX Security Symposium*. USENIX Association, 2020, pp. 1605–1622.

[25] Z. Zhang, A. Panda, L. Song, Y. Yang, M. W. Mahoney, P. Mittal, K. Ramchandran, and J. Gonzalez, "Neurotoxin: Durable backdoors in federated learning," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 26 429–26 446.

[26] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth, 1984.

[27] F. Wang, H. Zhu, R. Lu, Y. Zheng, and H. Li, "Achieve efficient and privacy-preserving disease risk assessment over multi-outsourced vertical datasets," *ACM Transactions on Intelligent Systems and Technology (TIST).*, vol. 19, no. 3, pp. 1492–1504, 2022.

[28] ——, "A privacy-preserving and non-interactive federated learning scheme for regression training with gradient descent," *Information Sciences.*, vol. 552, pp. 183–200, 2021.

[29] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 4004. Springer, 2006, pp. 486–503.

[30] C. Dwork, F. McSherry, K. Nissim, and A. D. Smith, "Calibrating noise to sensitivity in private data analysis," *Journal of Privacy and Confidentiality.*, vol. 7, no. 3, pp. 17–51, 2016.

[31] C. Dobraunig, M. Eichlseder, and F. Mendel, "Security evaluation of sha-224, sha-512/224, and sha-512/256," *Institute for Applied Information Processing and Communications, Graz University of Technology*, 2015.

[32] P. Paillier and D. Pointcheval, "Efficient public-key cryptosystems provably secure against active adversaries," in *ASIACRYPT*, ser. Lecture Notes in Computer Science, vol. 1716. Springer, 1999, pp. 165–179.

[33] S. D. Galbraith, S. W. Gebregiyorgis, and S. Murphy, "Algorithms for the approximate common divisor problem," *LMS Journal of Computation and Mathematics.*, p. 215, 2016.

[34] I. Yeh and C. Lien, "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients," *Expert systems with applications.*, vol. 36, no. 2, pp. 2473–2480, 2009.

[35] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decision Support Systems*, vol. 62, pp. 22–31, 2014.

[36] N. Chandran, N. Dasgupta, D. Gupta, S. L. B. Obbattu, S. Sekar, and A. Shah, "Efficient linear multiparty PSI and extensions to circuit/quorum PSI," in *CCS*. ACM, 2021, pp. 1182–1204.

[37] S. Hardy, W. Henecka, H. Ivey-Law, R. Nock, G. Patrini, G. Smith, and B. Thorne, "Private federated learning on vertically parti-

tioned data via entity resolution and additively homomorphic encryption," *CoRR*, vol. abs/1711.10677, 2017.
[38] J. Zhao, H. Zhu, F. Wang, R. Lu, H. Li, Z. Zhou, and H. Wan, "Accel: an efficient and privacy-preserving federated logistic regression scheme over vertically partitioned data," *Science China Information Sciences*, vol. 65, no. 7, pp. 1–2, 2022.
[39] Y. Wu, S. Cai, X. Xiao, G. Chen, and B. C. Ooi, "Privacy preserving vertical federated learning for tree-based models," *Proceedings of the VLDB Endowment.*, vol. 13, no. 11, pp. 2090–2103, 2020.
[40] Z. Tian, R. Zhang, X. Hou, L. Lyu, T. Zhang, J. Liu, and K. Ren, "Federboost: Private federated learning for gbdt," *IEEE Transactions on Dependable and Secure Computing*, 2023.
[41] Y. Zheng, S. Xu, S. Wang, Y. Gao, and Z. Hua, "Privet: A privacy-preserving vertical federated learning service for gradient boosted decision tables," *IEEE Trans. Serv. Comput.*, vol. 16, no. 5, pp. 3604–3620, 2023.
[42] H. Mahdikhani, R. Lu, Y. Zheng, J. Shao, and A. A. Ghorbani, "Achieving o ($\log^3$n) communication-efficient privacy-preserving range query in fog-based iot," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5220–5232, 2020.
[43] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *ASIACRYPT (1)*, ser. Lecture Notes in Computer Science, vol. 10624. Springer, 2017, pp. 409–437.
[44] A. Law, C. Leung, R. Poddar, R. A. Popa, C. Shi, O. Sima, C. Yu, X. Zhang, and W. Zheng, "Secure collaborative training and inference for xgboost," in *PPMLPCCS*. ACM, 2020, pp. 21–26.

Wei Xu was born in China, in 1999. He received the B.Eng. degree in information security from Xidian University, Xi'an, Shaanxi, China, in 2022. He is currently pursuing the M.A. degree in cyberspace security at Xidian University, Xi'an, Shaanxi, China. His research has been concerned with privacy preserving machine learning and applied cryptography.

Hui Zhu (Senior Member, IEEE) received the B.Sc. degree from Xidian University, Xi'an, Shaanxi, China, in 2003, the M.Sc. degree from Wuhan University, Wuhan, Hubei, China, in 2005, and the Ph.D. degree from Xidian University in 2009.

He was a Research Fellow with the School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, in 2013. Since 2016, he has been a Professor with the School of Cyber Engineering, Xidian University. His current research interests include applied cryptography, data security, and privacy.

Yandong Zheng (Member, IEEE) received the M.S. degree from the Department of Computer Science, Beihang University, China, in 2017, and received the Ph.D. degree from the Department of Computer Science, University of New Brunswick, Canada, in 2022.

Since 2022, she has been an Associate Professor with the School of Cyber Engineering, Xidian University. Her research interest includes cloud computing security and big data privacy.

Fengwei Wang (Member, IEEE) received his B.Sc. degree from Xidian University in 2016 and Ph.D. degree from Xidian University in 2021. In 2019, he was with the Faculty of Computer Science, University of New Brunswick as a visiting scholar.

Since 2021, he has been the lecturer with the School of Cyber Engineering, Xidian University, Xi'an, China. His research interests include the areas of applied cryptography, cyber security, and privacy.

Jiaqi Zhao was born in China, in 1997. He received the B.Eng. degree in information security from Xidian University, Xi'an, Shaanxi, China, in 2020, where he is currently pursuing the Ph.D. degree in cyberspace security. His research has been concerned with privacy-preserving machine learning.

Zhe Liu (Senior Member, IEEE) received the B.S. and M.S. degrees from Shandong University in 2008 and 2011, respectively, and the Ph.D. degree from the University of Luxembourg in 2015.

He is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include security, privacy, and cryptography solutions for the Internet of Things. He was a recipient of the prestigious FNR AwardsOutstanding Ph.D. Thesis Award in 2016, the ACM CHINA SIGSAC Rising Star Award in 2017, and the DAMO Academy Young Fellow in 2019. He has served as the General Co-Chair for CHES 2020 and CHES 2021.

Hui Li (Member, IEEE) received the B.Sc. degree from Fudan University in 1990 and the M.Sc. and Ph.D. degrees from Xidian University, China, in 1993 and 1998, respectively.

Since 2005, he has been a Professor with the School of Telecommunication Engineering, Xidian University. His research interests are in the areas of cryptography, wireless network security, information theory, and network coding.

Dr. Li has served as the TPC Co-Chair for ISPEC 2009 and IAS 2009; the General Co-Chair for E-Forensic 2010, ProvSec 2011, and ISC 2011; and the Honorary Chair for NSS 2014 and ASIACCS 2016.