

Bilateral Differentially Private Vertical Federated Boosted Decision Trees

Bokang Zhang
The Chinese University of Hong
Kong, Shenzhen
bokangzhang@link.cuhk.edu.cn

Zhikun Zhang
Zhejiang University
zhikun@zju.edu.cn

Haodong Jiang
The Chinese University of Hong
Kong, Shenzhen
haodongjiang@link.cuhk.edu.cn

Yang Liu
Bytedance Inc.
liyuyang.fromthu@bytedance.com

Lihao Zheng
The Chinese University of Hong
Kong, Shenzhen
lihaozheng@link.cuhk.edu.cn

Yuxiao Zhou
The Chinese University of Hong
Kong, Shenzhen
yuxiaozhou@link.cuhk.edu.cn

Shuaiting Huang
Zhejiang University
shuait_huang@zju.edu.cn

Junfeng Wu
The Chinese University of Hong
Kong, Shenzhen
junfengwu@cuhk.edu.cn

ABSTRACT

Federated learning is a distributed machine learning paradigm that enables collaborative training across multiple parties while ensuring data privacy. Gradient Boosting Decision Trees (GBDT), such as XGBoost, have gained popularity due to their high performance and strong interpretability. Therefore, there has been a growing interest in adapting XGBoost for use in federated settings via cryptographic techniques. However, it should be noted that these approaches may not always provide rigorous theoretical privacy guarantees and they often come with a high computational cost in terms of time and space requirements. In this paper, we propose a variant of vertical federated XGBoost with bilateral differential privacy guarantee: MaskedXGBoost. We build well-calibrated noise to perturb the intermediate information to protect privacy. The noise is structured with part of its ingredients in the null space of the arithmetical operation for splitting score evaluation in XGBoost, helping us achieve consistently better utility than other perturbation methods and relatively lower overhead than encryption-based techniques. We provide theoretical utility analysis and empirically verify privacy preservation. Compared with other algorithms, our algorithm's superiority in both utility and efficiency has been validated on multiple datasets.

1 INTRODUCTION

Big data is widely recognized to play an essential role in machine learning [48]. Traditionally, large datasets are aggregated from multiple sources and processed by a central server through a process known as *centralized learning*. However, this paradigm is becoming

increasingly problematic due to concerns over the unauthorized use and exploitation of users' personal data. *Federated learning* (FL) [38] addresses this challenge by transferring intermediate results of the training algorithm instead of the raw personal data. Based on how data is partitioned, FL can be roughly classified into two categories: Horizontal FL (HFL) and vertical FL (VFL). HFL, also known as sample-wise FL, targets the scenarios where participants' data have the same feature space but differ in samples [48]. VFL, or feature-wise FL, is applicable to cases where multiple datasets share the same sample ID space but differ in feature space [48]. While HFL has been extensively studied by the research community [2, 3, 37, 39, 41], less attention has been given to VFL. Existing VFL methods heavily rely on cryptography technologies such as homomorphic encryption and secure multiparty computation to calculate privacy-related intermediate results [6, 48].

Gradient Boosting Decision Trees (GBDT), such as XGBoost [4], are well-regarded for their high performance and interoperability. They have found widespread use in various industrial applications, including financial risk management [26, 36, 43] and online advertising [21, 35]. In these applications, training data and labels often belong to separate parties. For example, e-commerce companies may want to personalize product recommendations using financial data, like users' bank loans. In this scenario, label data is with the e-commerce company, and financial data is at the bank. This setup falls under VFL. Our paper focuses on extending the benefits of XGBoost to VFL settings with privacy guarantees.

Existing Solutions. Most of the existing studies on VFL-setting XGBoost require different types of encryption-based protocols: Homomorphic encryption (HE) [6, 25, 36] and secret-sharing [17], resulting in a large communication and computing overhead. Another line of research adopts the notion of *differential privacy* (DP) or *local differential privacy* (LDP) and performs the analysis on the perturbed data [34, 44, 46]. Le et al. [30] propose perturbing the gradient information at the label party, while Tian et al. [44] propose perturbing the feature information on the feature party side. Despite DP-based methods reducing the training time a lot, the model suffers accuracy loss from the injected noise. The work [30] also

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06...\$15.00
<https://doi.org/XXXXXXXX.XXXXXXX>

proposes to adopt *secure matrix multiplication* [27] in the private training of vertical federated XGBoost [30]. Secure matrix multiplication is a cryptographic technique that allows two parties to jointly compute the product of their matrices without revealing their inputs to each other. This approach offers low communication costs but without rigorous proof of privacy guarantee.

Our Contributions. Our proposed MaskedXGBoost ensures bilateral differential privacy guarantees for both parties during the training, and our method achieves consistently better model utility. We consider the evaluation of a splitting score, the major step requiring privacy guarantee in XGBoost, as the multiplication of a categorical matrix and a vector [30]. Our approach constructs well-calibrated noises to the sensitive information vector, which significantly enhances privacy while having a minimal impact on the utility of the data. Unlike previous research that only focused on analyzing the privacy leakage of one party holding the label, our approach employs differential privacy to ensure privacy preservation for both parties involved. Additionally, we build an attack model, trying to reveal any sensitive label information of users to check whether our protocol is secure enough to protect against label inference attacks. Our main contributions are as follows:

- **(C1) MaskedXGBoost.** We propose an algebra-based approach, called MaskedXGBoost, to achieve better model utility than perturbation methods and higher efficiency than HE methods. We conduct extensive experiments on six datasets to illustrate the improvement of utility and efficiency of MaskedXGBoost. It achieves 4.82× (Adult dataset) to 6.72× (Nomao dataset) training time improvement. We provide a theoretical analysis of the better utility property of MaskedXGBoost in Theorem 1.
- **(C2) New Idea for Designing Differential Private Mechanisms with Better Utility.** Rather than relying on conventional Gaussian mechanisms or other common differential privacy noise mechanisms, our approach involves designing noises tailored to the specific problem being solved to achieve differential privacy. Most of the noise is allocated to the null space of the arithmetical operation for splitting score evaluation in XGBoost, striking an acceptable balance between training utility and data privacy.
- **(C3) Differential Privacy Analysis for Both Parties.** We provide sufficient differential privacy proof for the so-called active party and passive party in the VFL setting. To the best of our knowledge, our bilateral privacy analysis is the first in the literature. In particular, we discover that different noise ingredients (see Section 4.2 for greater detail of noise calibration) affect the two parties of MaskedXGBoost differently. The energy of the whole noise should be considered for the active party’s privacy protection (see Theorem 2), while the energy ratio of different ingredients and the size of a training dataset are the main factors for the passive party’s privacy (see Theorem 3). We conduct the ablation study of the effect of the noise energy ratio on utility.
- **(C4) Empirical Privacy Evaluation.** We conduct empirical privacy evaluation by building a label inference attacker and an attribute inference attacker to verify the theoretical differential privacy results (see Theorem 2 and Theorem 3). Under a strict privacy budget, the attacker’s performance is poor (i.e., attack

acc 0.51 for AP and 0.52 for PP), demonstrating the security and privacy of our approach.

2 PRELIMINARIES

2.1 Vertical Federated Learning

Federated learning (FL) aims to facilitate the collaboration of multiple participants in contributing distinct training data to train a better model collectively. Based on how the data is partitioned, FL can be roughly classified into two categories [48]: *Horizontal federated learning* (HFL) and *vertical federated learning* (VFL). In this paper, we focus on VFL.

VFL. Vertical FL, also known as feature-wise FL, is suitable for situations where multiple datasets share the same sample ID space but have different feature spaces. For example, consider two different companies in a city, one is a bank, and the other is an e-commerce company. Their user sets are likely to contain most of the residents of the area, so the intersection of their user spaces is large. Since the bank has access to the user’s financial information, such as revenue and expenditure behavior, while the e-commerce company retains the user’s browsing and purchasing history, their feature spaces are significantly distinct. We want both parties to have a prediction model for product purchase based on user and product information.

2.2 XGBoost

Consider a dataset $\mathcal{D} = \{(x_i, y_i), x_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i \in \mathcal{I}\}$, where x_i denotes the feature vectors of the i^{th} instance in a feature space \mathcal{X} , y_i is the label of the i^{th} instance, and \mathcal{I} denotes the index set of instances. Let $n = |\mathcal{I}|$ be the total number of instances.

XGBoost. XGBoost is a boosting-based machine learning algorithm that ensembles a set of decision trees. Figure 1 shows an example of XGBoost. Next, we take a closer look at the XGBoost training. For a regression model $\psi(\cdot)$ consisting of T regression trees f_t , we choose a set \mathcal{F} of admissible tree models as follows

$$\hat{y}_i := \psi(x_i) = \sum_{t=1}^T f_t(x_i), f_t \in \mathcal{F}, x_i \in \mathcal{X}. \quad (1)$$

With any differentiable convex loss function, $l : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, capturing the disagreement between the labels y_i ’s and predicted outputs \hat{y}_i ’s, the objective function $L_0(\psi)$ for the model training process can be defined as

$$L_0(\psi) = \sum_{i=1}^n l(y_i, \hat{y}_i) + \sum_{t=1}^T \Omega(f_t), \quad (2)$$

where $\Omega(f_t) := \gamma L^{(t)} + \frac{1}{2} \lambda \left\| \mathbf{w}^{(t)} \right\|^2$ is the regularization on the model complexity of tree f_t to avoid overfitting, $L^{(t)}$ is the number of leaves and $\mathbf{w}^{(t)}$ is the leaf weight of regression tree f_t .

When we train a regression model, an iterative optimization method can be used to minimize the objective (2). At the t^{th} iteration, the previously constructed trees f_1, \dots, f_{t-1} remain unchanged, while a new tree f_t is trained for the first time and added to the regression model. Given the predicted output $\hat{y}_i^{(t-1)} =$

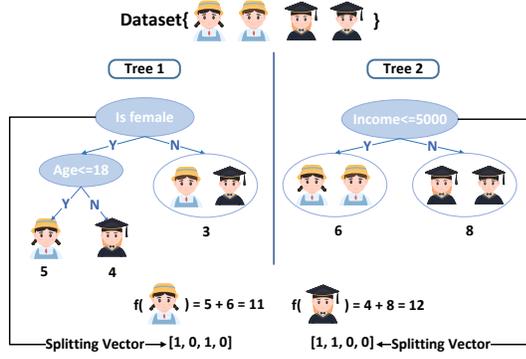


Figure 1: An example of XGBoost and splitting vector. The final prediction for instance is the sum of the predictions from each regression tree. At the first level of Tree1, we split the instances based on whether the individual is female or not, which generates the splitting vector [1, 0, 1, 0].

$\sum_{k=1}^{t-1} f_k(x_i)$ at the $(t-1)^{th}$ iteration, the objective at the t^{th} iteration is formulated as

$$L_0^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)}) + f_t(x_i) + \Omega(f_t). \quad (3)$$

To make the training process computationally efficient, rather than the objective itself, we usually minimize a second-order approximation of the objective function as follows

$$L^{(t)} = \sum_{i=1}^n \left(l(y_i, \hat{y}_i^{(t-1)}) + g_i^{(t)} f_t(x_i) + \frac{h_i^{(t)} f_t^2(x_i)}{2} \right) + \Omega(f_t), \quad (4)$$

where $g_i^{(t)} = \partial_{\hat{y}_i^{(t-1)}} l$ and $h_i^{(t)} = \partial_{\hat{y}_i^{(t-1)}}^2 l$ are the first and second derivative of the loss function at $\hat{y}_i^{(t-1)}$, respectively.

Let I_j denotes the index set of instances at the j^{th} leaf, i.e., $I_j = \{i | x_i \text{ belongs to } j^{th} \text{ leaf of } f_t\}$. Chen [4] has shown that the optimal weight of each leaf j is computed by

$$w_j^{(t),*} = - \frac{\sum_{i \in I_j} g_i^{(t)}}{\sum_{i \in I_j} h_i^{(t)} + \lambda}. \quad (5)$$

The following metric is often used to evaluate a *splitting candidate* at each node j (associated with instance set I_j) [4]:

$$L_{split}(I_{L,j}, I_{R,j}) = -\gamma + \frac{1}{2} \left(\frac{(\sum_{i \in I_{L,j}} g_i^{(t)})^2}{\sum_{i \in I_{L,j}} h_i^{(t)} + \lambda} + \frac{(\sum_{i \in I_{R,j}} g_i^{(t)})^2}{\sum_{i \in I_{R,j}} h_i^{(t)} + \lambda} - \frac{(\sum_{i \in I_j} g_i^{(t)})^2}{\sum_{i \in I_j} h_i^{(t)} + \lambda} \right), \quad (6)$$

where $I_{L,j}$ and $I_{R,j}$ are associated with the left and right child nodes, forming the dichotomy of the proposed splitting in question. We refer the readers to Appendix A for more details about XGBoost and the regression tree.

Splitting Vector. Next, we will illustrate XGBoost in another way in terms of the so-called splitting vector [30]. Let $d = [x_1, \dots, x_n]^T \in \mathbb{R}^n$ be the aggregation of a particular entry of the feature vectors for all instances and $s_i \in \mathbb{R}$ is the i^{th} splitting candidate for d to split the instance set into two separate sets at a node. The splitting operation $\text{Split}(d, s_i) : \mathbb{R}^n \times \mathbb{R} \rightarrow \{0, 1\}^n$

performs the comparison of each element in d with the splitting candidate s_i and outputs the i^{th} splitting vector $m_i \in \{0, 1\}^n$ as

$$m_i := \text{Split}(d, s_i) := [m_{1i}, \dots, m_{ni}]^T,$$

$$\text{where } m_{ji} = \begin{cases} 1, & x_j \leq s_i \\ 0, & x_j > s_i \end{cases}.$$

There corresponds a splitting operation to the splitting vector m_i for a splitting candidate s_i , which labels any instances as “1” if they are assigned to the left node and “0” to the right node. The following example depicts the functionality of the splitting operator and the splitting vector.

EXAMPLE 1. Consider a bank that holds data on many dimensions of its users, such as ages, genders, deposits, credit information, etc. These data can reflect the economic situation of users, which is helpful for predicting the purchasing powers of users. In Figure 1, the bank builds the XGBoost model using ages, genders, and income information. Suppose we were analyzing the effect of a user’s age. Let the feature vector be $d = [24, 25, 20, 22, 15, 17, 18, 16]^T$ representing the age data of a group of people. A splitting candidate s is chosen as $\text{Age} = 18$. The result of the splitting operator applied to the feature vector d and the candidate s is denoted by the following splitting vector

$$m = \text{Split}(d, s) = [0, 0, 0, 0, 1, 1, 1, 1]^T.$$

“1” means that person is no older than 18, and “0” means older than 18. The splitting operation by s (age = 18) is also depicted in Figure 1.

With the formulation of the splitting vector, the aggregated gradient and Hessian in each node can be computed by multiplying the splitting vector and gradient and Hessian vector. Let the gradient vector and Hessian vector of the n instances respectively be denoted as

$$g = [g_1, \dots, g_n]^T, h = [h_1, \dots, h_n]^T \in \mathbb{R}^n.$$

To evaluate a splitting candidate s , we need to know the aggregated gradients and Hessians for the split instances at the left and right nodes, termed g_s^L, h_s^L , and g_s^R, h_s^R , respectively, which are computed from m, g, h as follows:

$$g_s^L = m^T g, h_s^L = m^T h, g_s^R = G - g_s^L, h_s^R = H - h_s^L, \quad (7)$$

where $G := \sum_{i=1}^n g_i$ and $H := \sum_{i=1}^n h_i$. Then we can obtain the score L_{split}^s of the splitting rule candidate as

$$L_{split}^s = -\gamma + \frac{1}{2} \left(\frac{(g_s^L)^2}{(h_s^L) + \lambda} + \frac{(g_s^R)^2}{(h_s^R) + \lambda} - \frac{G^2}{H + \lambda} \right). \quad (8)$$

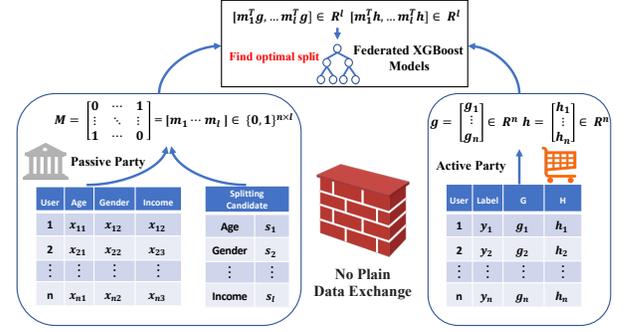
The essential of the training process for splitting at a node is to find the best splitting candidate s with the highest score in (8). Splitting is repeated from the newly constructed nodes until it reaches the maximum depth of the tree. The so-constructed nodes together form a whole tree structure.

2.3 Differential Privacy (DP) and Local Differential Privacy (LDP)

Differential Privacy. *Differential privacy* (DP) is a formal definition of privacy that guarantees the output of a data analysis does not depend significantly on a single individual’s data item. DP has been widely used in various fields, including control systems [24, 31], machine learning [1], data synthesis [45, 50, 53], etc.

Table 1: Summary of notations.

Notations	Descriptions
n	Number of the instances in a dataset
m_i	i^{th} splitting vector
M	Categorical matrix
g, h	Gradient and Hessian vector
g_i, h_i	Gradient and Hessian of i^{th} instance
W	Number of noise vectors for each m_i
b_{ij}	j^{th} noise vector for m_i
B_i	i^{th} noise matrix for m_i
$\langle g \rangle_i, \langle h \rangle_i$	Noised gradient and Hessian vector for m_i
$\epsilon_{\text{AP}}, \epsilon_{\text{PP}}$	Privacy budgets for active party and passive party

**Figure 2: Federated XGBoost problem setting.**

DEFINITION 1 (DIFFERENTIAL PRIVACY [14]). A randomized mechanism \mathcal{M} satisfies (ϵ, δ) -differential privacy $((\epsilon, \delta)$ -DP), where $\epsilon \geq 0$ and $0 \leq \delta \leq 1$, if for any two datasets inputs d and d' that differ in a single record and any measurable set O of \mathcal{M} 's outputs,

$$\Pr(\mathcal{M}(d) \in O) \leq \exp(\epsilon)\Pr(\mathcal{M}(d') \in O) + \delta.$$

Here ϵ is the *privacy budget*. To achieve (ϵ, δ) -differential privacy, the Gaussian and Laplace mechanisms are usually adopted by adding noise calibrated to the sensitivity of a function [13]. The Laplace mechanism can achieve strict ϵ -DP, while the Gaussian mechanism can only achieve (ϵ, δ) -DP. When $\delta = 0$, the mechanism \mathcal{M} satisfies pure (strict) differential privacy (pure DP), namely, ϵ -DP. When $\delta > 0$, the mechanism satisfies approximate (relaxed) differential privacy (approximate DP), namely, (ϵ, δ) -DP [13].

The following results are fundamental to the privacy analysis when multiple randomized functions are applied to a dataset.

LEMMA 1 (SEQUENTIAL COMPOSITION [14]). Let $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_k\}$ be a series of randomized mechanisms performed sequentially on a dataset. If \mathcal{M}_i provides (ϵ_i, δ_i) -DP, then \mathcal{M} provides (ϵ, δ) -DP with $\epsilon := \sum_{i=1}^k \epsilon_i$ and $\delta := \sum_{i=1}^k \delta_i$.

LEMMA 2 (PARALLEL COMPOSITION [14]). Let $\mathcal{M} = \{\mathcal{M}_1, \dots, \mathcal{M}_k\}$ be a series of mechanisms performed separately on disjoint subsets of the entire dataset. If \mathcal{M}_i provides (ϵ_i, δ_i) -DP, then \mathcal{M} provides (ϵ, δ) -DP with $\epsilon := \max\{\epsilon_1, \dots, \epsilon_k\}$, $\delta := \max\{\delta_1, \dots, \delta_k\}$.

For a sequential of k mechanisms $\mathcal{M}_1, \dots, \mathcal{M}_k$ satisfying (ϵ_i, δ_i) -DP respectively, the basic sequential composition result [14] shows that the privacy composes linearly, i.e., the sequential composition satisfies $(\sum_{i=1}^k \epsilon_i, \sum_{i=1}^k \delta_i)$ -DP. When $\epsilon_i = \epsilon$ and $\delta_i = \delta$, the advanced composition bound from [15] states that the composition satisfies $(\epsilon\sqrt{2k \log(1/\delta')} + k\epsilon(e^\epsilon - 1), k\delta + \delta')$ -DP, where $\delta' > 0$.

Local Differential Privacy. A stronger notion of DP is *local differential privacy* (LDP), which perturbs the individual's data locally to protect private information [9, 10, 47, 52]. LDP is a model of differential privacy with the added restriction that even if an adversary has access to the personal responses of an individual in the database, that adversary is still unable to learn too much about the user's personal data. This is contrasted with DP, which incorporates a central aggregator with access to the raw data.

DEFINITION 2 (LOCAL DIFFERENTIAL PRIVACY [16]). A randomized mechanism \mathcal{M} satisfies (ϵ, δ) -local differential privacy $((\epsilon, \delta)$ -LDP), where $\epsilon \geq 0$ and $0 \leq \delta \leq 1$, if for any two of input d, d' , and any measurable set O of \mathcal{M} 's outputs,

$$\Pr(\mathcal{M}(d) \in O) \leq \exp(\epsilon)\Pr(\mathcal{M}(d') \in O) + \delta.$$

2.4 Notations

Frequently used notations are summarised in Table 1.

3 PROBLEM STATEMENT AND EXISTING SOLUTIONS

3.1 Problem Statement

Federated XGBoost. In this paper, we focus on VFL-setting XGBoost. We call a party that holds both feature data and the class label *active party* (AP) and call a party that merely holds the feature data *passive party* (PP). We consider that AP and PP want to train an XGBoost model jointly. The essential part of the training state is computing the splitting score (8) and then finding the optimal splitting candidate. Assume that at each node, PP has l splitting candidates, through each splitting vector $m_i \in \mathbb{R}^n$, the categorical matrix M is defined as $M = [m_1, \dots, m_l] \in \mathbb{R}^{n \times l}$. According to (7), AP has g, h , while PP owns M and they desire to compute $M^T g, M^T h$ securely. Gradient and Hessian vectors in plain text cannot be sent directly, otherwise, users' label information from AP will be seriously leaked [6]. Figure 2 illustrates the problem setting. With $M^T g, M^T h$ known, the optimal splitting candidate can be found and the federated XGBoost model can be built step by step.

REMARK 1. *Multi-party private set intersection [29] is employed to determine the intersection of common databases among participants securely, which is orthogonal to our paper. The primary challenge in our setting lies in how to jointly conduct the training process among participants with the aligned database.*

Threat Model and Design Goals. In common with many other works in the federated setting [6, 32, 37, 51], we assume an honest-but-curious model, where the clients do not trust others with their raw data. Given the above setting, we aim to propose a federated XGBoost protocol with the following design goals:

- **G1: Utility.** Its utility should be close to centralized learning, which is to pool all data into a centralized server.

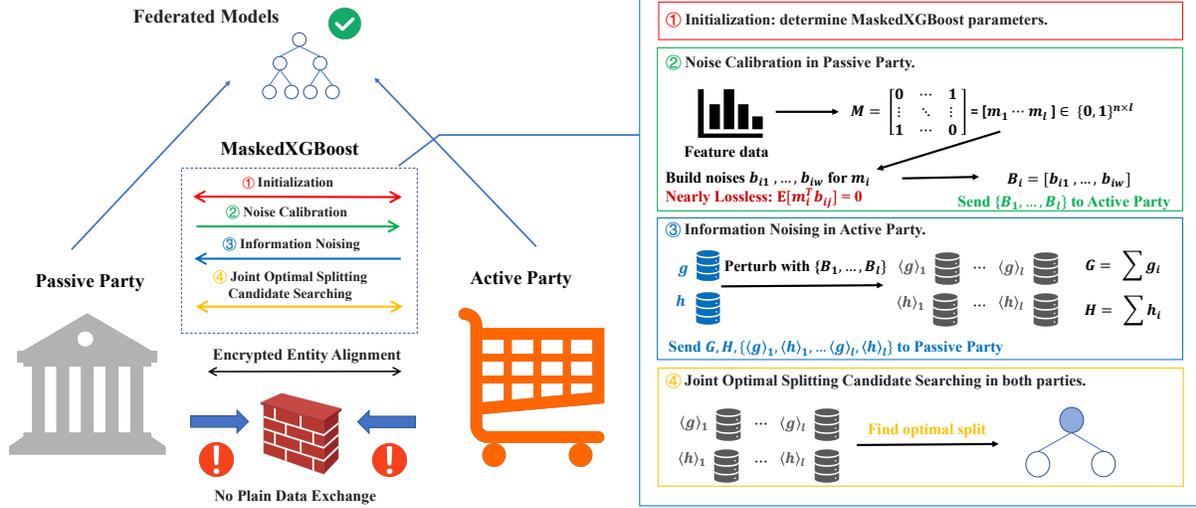


Figure 3: MaskedXGBoost overview. MaskedXGBoost is composed of four steps: initialization, noise calibration, information noising, and joint optimal splitting candidate searching. AP and PP first determine the necessary algorithm parameters in the initialization step. In the noise calibration step, PP builds well-calibrated noise according to the categorical matrix which ensures the consistently better property of MaskedXGBoost. In the information noising step, AP uses the noises PP sent to perturb the sensitive information g and h , then AP sends the information back to PP. Finally, AP and PP jointly find the optimal splitting candidate. Then, splitting continues from the newly constructed nodes until it reaches the maximum depth of the tree. AP and PP stand for active party and passive party respectively.

- **G2: Privacy.** Its privacy should be close to local training, i.e., each participant trains with its local data only. To achieve this, all data being transmitted should be protected either by cryptographic technology or differential privacy.
- **G3: Efficiency.** Its efficiency should be close to traditional distributed ML [4, 20, 28], i.e., the number of cryptographic operations should be minimized.

3.2 Existing Solutions

HE Methods. Current approaches that rely on Homomorphic Encryption (HE) are proficient in safeguarding privacy but often lack efficiency [5, 6]. At the same time, these methods lack the protection of PP’s private feature information. These methods can satisfy G1 (Utility) and G2 (Privacy) of only one party AP, while they cannot satisfy G2 (Privacy) of the other party PP and G3 (Efficiency).

DP Methods. The methods [30, 44] using DP have a great improvement in efficiency, but the utility is greatly disturbed. The work [46] considers the VFL settings for decentralized labels, utilizing secure aggregation and differential privacy to protect the privacy. Since only AP owns the label in our two-party VFL setting, this method is equivalent to adding noise directly to the gradient information. These methods can satisfy G2 (Privacy) of both parties and G3 (Efficiency), with serious sacrifice of G1 (Utility).

Other Methods. In the splitting vector view of XGBoost, Le et al. [30] proposed FedXGBoost-SMM. FedXGBoost-SMM turns the federated XGBoost into a secure matrix multiplication problem, and its key idea is to build kernel vectors as noise vectors. FedXGBoost-SMM can not provide a differential privacy guarantee and finding kernel vectors for a large categorical matrix is still time-consuming.

FedXGBoost-SMM can satisfy G1 (Utility) with the sacrifice of G2 (Privacy) and G3 (Efficiency).

4 OUR PROPOSAL: MASKEDXGBOOST

4.1 Overview

The general idea of MaskedXGBoost is that PP generates well-calibrated noises according to the categorical matrix M , and AP uses these noises to perturb sensitive vectors g, h , and sends the masked vectors back to PP. Then PP multiplies the categorical matrix and the masked vectors to find an optimal splitting candidate. The whole process of the protocol includes four steps: Initialization, noise calibration, information noising, and joint optimal splitting candidate searching. Figure 3 illustrates the main steps of MaskedXGBoost.

(S1) Initialization. Before the protocol begins in earnest, the two parties negotiate a level of differential privacy guarantee. Then according to this privacy setting, AP and PP can get the suitable parameters σ_1, σ_2, C, W which will be applied to the concrete procedures in the following steps, with the guidance of Theorem 1, Theorem 2, and Theorem 3, which will be introduced in more detail in Section 5. Before the whole training, AP randomly initializes the predicted labels and computes the initial gradients and Hessians [6].

(S2) Noise Calibration in PP. In this step, PP builds noises calibrated to the categorical matrix M and then sends the noises to AP. This step will be introduced in detail in Section 4.2.

(S3) Information Noising in AP. After receiving the noises PP sent, AP uses these noises to perturb the sensitive information and then sends them to PP for the next step. This step will be introduced in detail in Section 4.3.

Algorithm 1: Noise Calibration

1 **Input:** Private categorical matrix $M = [m_1, \dots, m_l] \in \mathbb{R}^{n \times l}$,
 $m_i \in \{0, 1\}^n$, MaskedXGBoost parameters $\sigma_1, \sigma_2, W > 0$

2 **Output:** Noise matrices set $\{B_1, \dots, B_l\}$

3 **Procedures:**

- 1: **for** $i = 1$ to l **do**
- 2: **for** $j = 1$ to W **do**
- 3: $u_{ij}, v_{ij}, r_{ij} \leftarrow \{0\}^n$
- 4: Generate $p_1, \dots, p_{n_{A,i}} \sim \mathcal{N}(0, \sigma_1^2)$,
 $q_1, \dots, q_{n_{I,i}} \sim \mathcal{N}(0, 2\sigma_1^2)$, $r_{ij} \sim \mathcal{N}(0, \sigma_2^2 I)$
- 5: **for** $k = 1$ to $n_{A,i}$ **do**
- 6: **if** $k = 1$ **then**
- 7: $[u_{ij}]_{\mathcal{A}(m_i)_k} = p_1 - p_{n_{A,i}}$
- 8: **else**
- 9: $[u_{ij}]_{\mathcal{A}(m_i)_k} = p_k - p_{k-1}$
- 10: **end if**
- 11: **end for**
- 12: **for** $k = 1$ to $n_{I,i}$ **do**
- 13: $[v_{ij}]_{\mathcal{I}(m_i)_k} = q_k$
- 14: **end for**
- 15: $b_{ij} \leftarrow u_{ij} + v_{ij} + r_{ij}$
- 16: **end for**
- 17: $B_i \leftarrow [b_{i1}, \dots, b_{iW}]$
- 18: **end for**

Return: Noise matrices set $\{B_1, \dots, B_l\}$

(S4) Joint Optimal Splitting Candidate Searching. With the information AP sent, PP can evaluate splitting candidates and then find the optimal splitting candidate according to (8). PP is requested to reveal the corresponding splitting score and operation. AP finds the best splitting candidate over AP and PP. Then AP constructs new tree nodes and repeats the process with the new set of users.

The MaskedXGBoost algorithm repeats steps **S2-S4** until a tree reaches the maximum depth, and builds new trees till the maximum training rounds. The predicted labels are updated based on the evolving tree structure, and gradients and Hessians are updated accordingly.

4.2 Noise Calibration

In this step, PP builds noise vectors according to the categorical matrix M . We combine the Gaussian mechanism to construct orthogonal vectors of each splitting vector m_i and then take these orthogonal vectors as noise vectors.

Active Set and Inactive Set. To facilitate the presentation, we give the definition of the active set and inactive set. The *active set* refers to the index set of a splitting vector that has a value of “1”, while the *inactive set* refers to the index set that has a value of “0”. For each splitting vector $m_i \in \{0, 1\}^n$, we define the active set $\mathcal{A}(m_i)$ and the inactive set $\mathcal{I}(m_i)$ as follows:

$$\begin{aligned} \mathcal{A}(m_i) &:= \{j \in \mathcal{I} \mid m_{ij} = 1\}. \\ \mathcal{I}(m_i) &:= \mathcal{I} \setminus \mathcal{A}(m_i) = \{j \in \mathcal{I} \mid m_{ij} = 0\}. \end{aligned}$$

We denote $n_{A,i} = |\mathcal{A}(m_i)|$ and $n_{I,i} = |\mathcal{I}(m_i)|$. The active set includes all the nonzero entries of a splitting vector, while the inactive

set encapsulates the zero. For noise calibration, We will combine differently structured noises tailored to the active and inactive sets, namely the active and inactive noises.

Noise Calibration. Algorithm 1 describes the procedure of building noises. For each m_i , we build a noise matrix $B_i = [b_{i1}, \dots, b_{iW}]$, where $b_{ij} \in \mathbb{R}^n$ consists of three parts, *active noise* u_{ij} , *inactive noise* v_{ij} , and *disturbing noise* r_{ij} , i.e. $b_{ij} = u_{ij} + v_{ij} + r_{ij}$. The active noise perturbs the bits of the active set in the gradient and Hessian vector. It is spatially colored as it is in the null space of m_i . The inactive noise perturbs the bits of the inactive set in the gradient and Hessian vector freely. The disturbing noise perturbs all the bits of the gradient and Hessian vector together. It is critical for the privacy protection of PP’s categorical matrix. Next, we introduce how the three kinds of noises are constructed.

Active Noise u_{ij} . Generate i.i.d. random numbers $p_1, \dots, p_{n_{A,i}}$ following Gaussian distribution $\mathcal{N}(0, \sigma_1^2)$. We use $\mathcal{A}(m_i)_k$ to denote the k^{th} element in the set $\mathcal{A}(m_i)$. Firstly, we initialize $u_{ij} \in \mathbb{R}^n$ as 0, and then assign values to u_{ij} iteratively:

$$[u_{ij}]_{\mathcal{A}(m_i)_k} = \begin{cases} p_1 - p_{n_{A,i}}, & k = 1 \\ p_k - p_{k-1}, & k = 2, \dots, n_{A,i} \end{cases}.$$

It can be checked that $m_i^\top u_{ij} = 0$ since the elements in the active noise u_{ij} cancel each other out. The active noise does not affect the calculation result of $m_i^\top g$ and $m_i^\top h$ if used for information noising, indicating that this part of the noise is lossless.

Inactive Noise v_{ij} . Generate i.i.d. random numbers $p_1, \dots, p_{n_{I,i}}$ following Gaussian distribution $\mathcal{N}(0, 2\sigma_1^2)$. We use $\mathcal{I}(m_i)_k$ to denote the k^{th} element in the set $\mathcal{I}(m_i)$. First we initialize the inactive noise as $v_{ij} = 0$, then assign values to v_{ij} iteratively:

$$[v_{ij}]_{\mathcal{I}(m_i)_k} = q_k, k = 1, \dots, n_{I,i}.$$

It is immediately known that $m_i^\top v_{ij} = 0$ because v_{ij} only operates on the bits in the inactive set, which correspond to the bits in m_i with a value of 0. Similarly to the active noise, this implies that the inactive noise does not affect the calculation result of $m_i^\top g$ and $m_i^\top h$ too if used for information noising, revealing the lossless property of the inactive noise.

Disturbing Noise r_{ij} . The disturbing noise is generated much more easily. We generate r_{ij} free from a multidimensional Gaussian distribution, $r_{ij} \sim \mathcal{N}(0, \sigma_2^2 I_n)$. It holds that

$$\mathbb{E}[m_i^\top r_{ij}] = 0, \text{Var}(m_i^\top r_{ij}) = n_{A,i} \sigma_2^2.$$

This is because although r_{ij} operates on all the bits, only the bits in the active set will perturb the outcomes effectively.

Repeating the noise generation process above for W times, PP builds b_{i1}, \dots, b_{iW} as a whole noise matrix B_i , i.e. $B_i = [b_{i1}, \dots, b_{iW}] \in \mathbb{R}^{n \times W}$. Similarly, PP can generate B_1, \dots, B_l according to l splitting vectors m_1, \dots, m_l . Then PP sends the noise matrices set $\{B_1, \dots, B_l\}$ to AP. The following example can visualize the noise generation procedure.

EXAMPLE 2. We consider the setting in Example 1, assuming the splitting candidate $\text{Age} = 18$ is the i^{th} splitting candidate. After analyzing the users’ age information, PP builds the splitting vector $m_i = [0\ 0\ 0\ 0\ 1\ 1\ 1\ 1]^T$. The active set $\mathcal{A}(m_i)$ is $\{5, 6, 7, 8\}$, inactive set $\mathcal{I}(m_i)$ is $\{1, 2, 3, 4\}$. Generate i.i.d. random variables p_1, \dots, p_4 according to $\mathcal{N}(0, \sigma_1^2)$ and q_1, \dots, q_4 according to $\mathcal{N}(0, 2\sigma_1^2)$. Next we construct the first noise b_{i1} , which means constructing u_{i1}, v_{i1}, r_{i1} .

Algorithm 2: Information Noising

1 **Input:** Noise matrices set $\{B_1, \dots, B_l\}$, Private data $g, h \in \mathbb{R}^n$, MaskedXGBoost parameter $C > 0$

2 **Output:** $\{\langle g \rangle_i, \langle h \rangle_i\}_{i \in \{1, \dots, l\}}, G, H$

3 **Procedures:**

- 1: **for** $i = 1$ to l **do**
- 2: Generate c_{i1}, \dots, c_{iW} and d_{i1}, \dots, d_{iW} satisfy $\sum_{k=1}^W c_{ik}^2 = \sum_{k=1}^W d_{ik}^2 = C$
- 3: $\langle g \rangle_i \leftarrow g + \sum_{k=1}^W c_{ik} b_{ik}, \langle h \rangle_i \leftarrow h + \sum_{k=1}^W d_{ik} b_{ik}$
- 4: **end for**
- 5: $G \leftarrow \sum_{i=1}^n g_i, H \leftarrow \sum_{i=1}^n g_i$

Return: $\{\langle g \rangle_i, \langle h \rangle_i\}_{i \in \{1, \dots, l\}}, G, H$

$$u_{i1} = \begin{bmatrix} 0 \\ p \end{bmatrix}, v_{i1} = \begin{bmatrix} q \\ 0 \end{bmatrix}, r_{i1} \sim \mathcal{N}(0, \sigma_2^2 I)$$

where $p = [p_1 - p_4, p_2 - p_1, p_3 - p_2, p_4 - p_3]^\top$ and $q = [q_1, q_2, q_3, q_4]^\top$. Overall, $b_{i1} = u_{i1} + v_{i1} + r_{i1}$. Now We use a numerical simulation to illustrate this procedure. Let $\sigma_1^2 = 1, \sigma_2^2 = 0.1$. Numerically generated noises are $p_1 = 0.9109, p_2 = -0.2397, p_3 = 0.1810, p_4 = 0.2442$, and

$$\begin{aligned} u_{i1} &= [0, 0, 0, 0, 0.6667, -1.1506, 0.4207, 0.0632]^\top, \\ v_{i1} &= [-0.9613, 1.6737, 5.0767, -2.6467, 0, 0, 0, 0]^\top, \\ r_{i1} &= [0.0138, -0.1907, -0.0365, -0.0848, \\ &\quad -0.0765, -0.1128, 0.0078, 0.2107]^\top. \end{aligned}$$

The noise vector is

$$\begin{aligned} b_{i1} &= [-0.9475, 1.4830, 5.0402, -2.7315, \\ &\quad 0.5902, -1.2634, 0.4285, 0.2739]^\top. \end{aligned}$$

Thus we can verify that

$$m_i^\top u_{i1} = 0, m_i^\top v_{i1} = 0, m_i^\top r_{i1} = 0.0292, m_i^\top b_{i1} = 0.0292$$

We see that the active noise u_{i1} and inactive noise v_{i1} are lossless. and that b_{i1} follows a Gaussian distribution $\mathcal{N}(0, \Sigma)$, where $\Sigma = \text{diag}(\Sigma_1, \Sigma_2)$ with $\Sigma_1 = 2.1I_4$ and

$$\Sigma_2 = \begin{bmatrix} 2.1 & -1 & 0 & -1 \\ -1 & 2.1 & -1 & 0 \\ 0 & -1 & 2.1 & -1 \\ -1 & 0 & -1 & 2.1 \end{bmatrix}$$

4.3 Information Noising

Algorithm 2 describes the process of noising sensitive information. Concretely, AP has the sensitive information gradient and Hessian $g, h \in \mathbb{R}^n$ that needs to be perturbed. We use $\langle g \rangle_i$ and $\langle h \rangle_i$ to denote the noised gradient and Hessian for i^{th} splitting candidate, respectively. AP needs to generate $\langle g \rangle_i, \langle h \rangle_i \in \mathbb{R}^n$ for each splitting candidate s_i .

Next we introduce how $\langle g \rangle_i, \langle h \rangle_i$ are generated with $B_i = [b_{i1}, \dots, b_{iW}] \in \mathbb{R}^{n \times W}$. AP generates $2W$ random numbers

Algorithm 3: Split Finding

1 **Input:**

- Dataset \mathcal{D} , with $|\mathcal{D}| = n$
- AP: Private data $g, h \in \mathbb{R}^n$
- PP: Private categorical matrix $M = [m_1, \dots, m_l] \in \{0, 1\}^{n \times l}$

Output of protocols: optimal splitting candidate s^*

Procedures:

- 1: **PP:** $\{B_1, \dots, B_l\} \xleftarrow{\text{Algorithm 1}} \text{Noise Calibration}(M)$
- 2: **PP:** Transmit $\{B_1, \dots, B_l\}$ to AP.
- 3: **AP:** $\{\langle g \rangle_i, \langle h \rangle_i\}_{i \in \{1, \dots, l\}}, G, H \xleftarrow{\text{Algorithm 2}} \text{Information Noising}(\{B_1, \dots, B_l\})$
- 4: **AP:** Transmit $\{\langle g \rangle_i, \langle h \rangle_i\}_{i \in \{1, \dots, l\}}, G, H$ to PP.
- 5: **PP:** /* Compute the aggregated gradients and Hessians */
- 6: $\{g_{s_i}^L, h_{s_i}^L\} \leftarrow \{m_i^\top \langle g \rangle_i, m_i^\top \langle h \rangle_i\}$
- 7: $L^* \leftarrow -\infty, s^* \leftarrow 0$
- 8: **for** $i = 1$ to l **do**
- 9: $g_{s_i}^R \leftarrow G - g_{s_i}^L, h_{s_i}^R \leftarrow H - h_{s_i}^L$
- 10: $L \leftarrow \frac{1}{2} \left(\frac{(g_{s_i}^L)^2}{h_{s_i}^L + \lambda} + \frac{(g_{s_i}^R)^2}{h_{s_i}^R + \lambda} - \frac{G^2}{H + \lambda} \right)$
- 11: **if** $L^* < L$ **then**
- 12: $L^* \leftarrow L, s^* \leftarrow s_i$
- 13: **end if**
- 14: **end for**
- 15: Transfer L^* to AP, save s^*
- 16: **AP:** Finds the optimal splitting candidate s^* over AP and PP.

c_{i1}, \dots, c_{iW} and d_{i1}, \dots, d_{iW} with a constraint that

$$\sum_{k=1}^W c_{ik}^2 = \sum_{k=1}^W d_{ik}^2 = C.$$

Then we perturb g, h as

$$\langle g \rangle_i := g + \sum_{k=1}^W c_{ik} b_{ik}, \langle h \rangle_i := h + \sum_{k=1}^W d_{ik} b_{ik}. \quad (9)$$

So $\langle g \rangle_1, \dots, \langle g \rangle_l$ and $\langle h \rangle_1, \dots, \langle h \rangle_l$ can be generated iteratively. If PP wants to evaluate each splitting candidate, PP needs to know the sum of gradients and Hessians G and H in (8). So after noising, AP compute

$$G = \sum_{i=1}^n g_i, H = \sum_{i=1}^n g_i.$$

Finally AP sends $\{\langle g \rangle_i, \langle h \rangle_i\}_{i \in \{1, \dots, l\}}, G, H$ to PP.

Intuitively, we can observe that increasing the value of C leads to a higher perturbation being added to g and h , resulting in stronger privacy protection. We constrain the sum of c_{ik}^2 and d_{ik}^2 to control the level of perturbation of the information. This will be analyzed in detail in Section 5.

4.4 Putting Things Together

With the information AP sent at the end of the information noising step, PP evaluates splitting candidates and then finds the optimal splitting candidate according to (8). Algorithm 3 describes

the procedure of optimal split finding. PP is requested to reveal the corresponding splitting score and splitting operation. AP finds the best splitting candidate according to the splitting score. AP then constructs new nodes and repeats the process with the new set of users. In the next section, we will discuss how to set the MaskedXGBoost's parameters σ_1, σ_2, C, W considering both model utility and privacy guarantee.

REMARK 2. *Our algorithm can be adapted to the GBDT algorithm. GBDT is similar to XGBoost, but it only needs gradients to calculate the splitting score. We can protect the gradients by transmitting the noised gradients in Algorithm 2 for the adaption of GBDT.*

5 THEORETICAL ANALYSIS

5.1 Utility Analysis

In this section, we show that MaskedXGBoost is consistently better. Similar to the related work that adds noise to the adjusted score function [30], the performance of MaskedXGBoost is related to the disturbance of the splitting scoring function (7). A higher level of disturbance in the scoring function increases the likelihood of failing to identify an optimal splitting candidate, which in turn leads to sub-optimal performance.

We show this by evaluating i^{th} splitting candidate as an example. Because of the particularity of the added noise in Algorithm 1, it can be checked that

$$m_i^\top \langle g \rangle_i = m_i^\top g + \sum_{k=1}^W c_{ik} r_{ik}, \quad m_i^\top \langle h \rangle_i = m_i^\top h + \sum_{k=1}^W d_{ik} r_{ik}, \quad (10)$$

indicating that $m_i^\top \langle g \rangle_i$ and $m_i^\top \langle h \rangle_i$ are merely disturbed by r_{ik} , making a better utility protocol possible. Notice that the splitting score function becomes random due to the noised aggregated gradients and Hessians $\langle g \rangle_i, \langle h \rangle_i$. Define the newly noised splitting score function $\langle L_{split}^{s_i} \rangle$ as

$$\langle L_{split}^{s_i} \rangle = -\gamma + \frac{1}{2} \left(\frac{(\langle g_i^L \rangle)^2}{\langle h_i^L \rangle + \lambda} + \frac{(\langle g_i^R \rangle)^2}{\langle h_i^R \rangle + \lambda} - \frac{G^2}{H + \lambda} \right).$$

where $\langle g_i^L \rangle = m_i^\top \langle g \rangle_i, \langle h_i^L \rangle = m_i^\top \langle h \rangle_i$. The utility of MaskedXGBoost can be evaluated by the following concentration analysis.

THEOREM 1. *Denote $\kappa := \sqrt{n_{A,i}} C \sigma_2$, there exists an upper bound $U(\alpha, \kappa)$ such that for any deviation $\alpha > 0$:*

$$\Pr \left(\left| \langle L_{split}^{s_i} \rangle - L_{split}^{s_i} \right| \geq \alpha \right) \leq U(\alpha, \kappa)$$

and $\lim_{\kappa \rightarrow 0} U(\alpha, \kappa) = 0$. In particular, for $\alpha \leq \min \left(\frac{(g_i^L)^2}{(h_i^L) + \lambda}, \frac{(g_i^R)^2}{(h_i^R) + \lambda} \right)$, $U(\alpha, \kappa)$ is given by (11).

The proof can be found in Appendix B. From (10), we see that MaskedXGBoost's performance is related to C and σ_2 , and it has nothing to do with σ_1 . Theorem 1 supports this hypothesis well. We claim that MaskedXGBoost achieves better utility in the sense that we can choose σ_2 and C to make the error bound $U(\alpha, \kappa)$ for estimating the splitting score sufficiently small. In Section 6.7, we also test the effect of σ_1 and σ_2 on MaskedXGBoost's performance numerically. Next, we introduce how to set reasonable parameters C, σ_1 , and σ_2 so that MaskedXGBoost satisfies the differential privacy guarantee for both parties of federated scenarios.

5.2 Privacy Analysis: Differential Privacy at Active Party Side

According to our protocol described in Section 4, AP needs to send the masked $\langle g \rangle_i, \langle h \rangle_i$ to PP, which can be used to discover the class label information [6]. So we need to analyze the privacy preservation of AP.

For each splitting vector m_i , AP uses the noises b_{i1}, \dots, b_{iW} received from PP to perturb g, h according to Algorithm 2. This is equivalent to adding a noise vector η_i directly to $g, h, \eta_i \sim \mathcal{N}(0, \Sigma)$, where the submatrix with rows and columns indexed by $\mathcal{A}(m_i)$ is given as

$$\Sigma_{\mathcal{A}(m_i)} = \begin{bmatrix} C(2\sigma_1^2 + \sigma_2^2) & -C\sigma_1^2 & 0 & \cdots & 0 & -C\sigma_1^2 \\ -C\sigma_1^2 & \ddots & \ddots & \ddots & & 0 \\ 0 & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & & & & \ddots & -C\sigma_1^2 \\ -C\sigma_1^2 & 0 & \cdots & 0 & -C\sigma_1^2 & C(2\sigma_1^2 + \sigma_2^2) \end{bmatrix} \quad (12)$$

and the submatrix of all the other entries is given as

$$\Sigma_{\mathcal{I}(m_i)} = C(2\sigma_1^2 + \sigma_2^2)I. \quad (13)$$

Notice that $\Sigma_{\mathcal{A}(m_i)}$ is a circulant matrix [7]. We employ LDP to evaluate the privacy preservation of our protocol.

To perform the LDP analysis, it is essential to bound the sensitivity of the query function. In our context, this requires establishing upper bounds for both the gradients and Hessians. For binary classification problems, the gradient and Hessian are bounded, i.e., there exists a $\mu > 0$ such that $|g_i|, |h_i| < \mu/2$, and μ can be conveniently derived. Specifically, the loss function is of the form $l(y_i, \hat{y}_i) = y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$ (i.e., binary cross-entropy). We have gradients $g_i \in [-\frac{1}{2}, -\frac{1}{e+1}] \cup [\frac{1}{2}, \frac{e}{e+1}]$ and Hessians $h_i \in [0, \frac{1}{4}]$; thus $\mu = \frac{2e}{e+1}$. For other problems (e.g., regression problems), the loss functions may have unbounded gradient and Hessian. We may use gradient clipping (similar to DP-SGD [1]) to obtain bounded gradients and Hessians and then derive μ .

THEOREM 2. *Recall that $n_{A,i}$ and $n_{I,i}$ are the sizes of the active set and the inactive set of the splitting vector m_i respectively, MaskedXGBoost is $(\epsilon_{AP}, \delta_{AP})$ -local differentially private, where $\epsilon_{AP} > 0, 0 < \delta_{AP} < 1$, for AP sending $\langle g \rangle_i, \langle h \rangle_i$ each time if $\frac{n_{I,i}\mu^2}{C(2\sigma_1^2 + \sigma_2^2)} + \frac{n_{A,i}\mu^2}{C\sigma_2^2} \leq 2 \left(\epsilon_{AP} - 2 \ln \delta_{AP} - 2\sqrt{\ln \delta_{AP} (\ln \delta_{AP} - \epsilon_{AP})} \right)$, where μ characterizes the value range of the gradients and Hessians.*

The proof can be found in Appendix C. By Theorem 2, we observe that the noise energy required is related to the specific splitting vector m_i , enabling our algorithm to adaptively adjust the noise scale throughout the training process. In the following section, we analyze the impact of σ_1 and σ_2 on PP's privacy leakage.

5.3 Privacy Analysis: Differential Privacy at Passive Party Side

Upon receiving the noise matrices B_1, \dots, B_l , privacy leakage occurs for PP. Specifically, with a given noise vector b_{ij} , AP aims to infer m_i .

$$\begin{aligned}
U(\alpha, \kappa) &= \int_0^\infty \left(\Phi\left(\frac{-\sqrt{\beta_L}t + \mu_X}{\kappa}\right) + \Phi\left(\frac{-\sqrt{\beta_L}t - \mu_X}{\kappa}\right) + \Phi\left(\frac{\sqrt{\beta'_L}t - \mu_X}{\kappa}\right) + \Phi\left(\frac{\sqrt{\beta'_L}t + \mu_X}{\kappa}\right) \right) \frac{1}{\kappa} \phi\left(\frac{t - \mu_Y}{\kappa}\right) dt \\
&+ \int_0^\infty \left(\Phi\left(\frac{-\sqrt{\beta_R}t + \mu_Z}{\kappa}\right) + \Phi\left(\frac{-\sqrt{\beta_R}t - \mu_Z}{\kappa}\right) + \Phi\left(\frac{\sqrt{\beta'_R}t - \mu_Z}{\kappa}\right) + \Phi\left(\frac{\sqrt{\beta'_R}t + \mu_Z}{\kappa}\right) \right) \frac{1}{\kappa} \phi\left(\frac{t - \mu_W}{\kappa}\right) dt \\
&+ \Phi(-\mu_Y/\kappa) + \Phi(-\mu_W/\kappa) \\
\beta_L &:= \frac{(g_i^L)^2}{(h_i^L) + \lambda} + \alpha, \quad \beta'_L := \frac{(g_i^L)^2}{(h_i^L) + \lambda} - \alpha, \quad \beta_R := \frac{(g_i^R)^2}{(h_i^R) + \lambda} + \alpha, \quad \beta'_R := \frac{(g_i^R)^2}{(h_i^R) + \lambda} - \alpha \\
\mu_X &:= (g_i^L), \quad \mu_Y := (h_i^L) + \lambda, \quad \mu_Z := (g_i^R), \quad \mu_W := (h_i^R) + \lambda
\end{aligned} \tag{11}$$

with standard normal density ϕ and cumulative distribution function Φ .

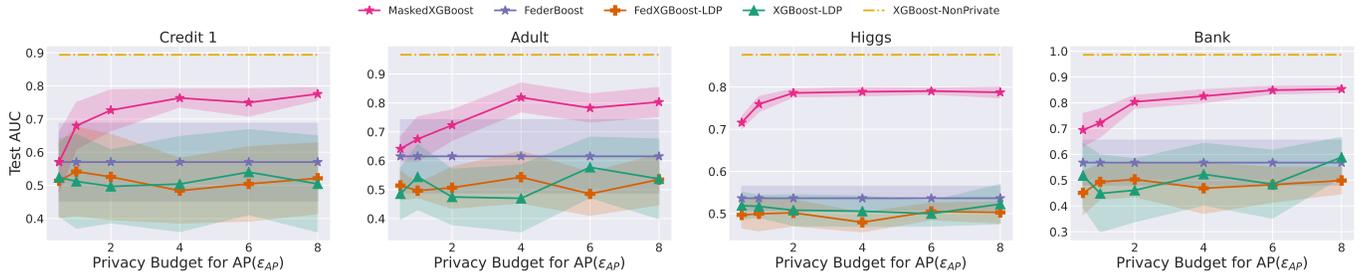


Figure 4: Utility of different methods for different privacy budgets.

m_i reveals the privacy of the user's feature data in PP. Recall Example 1, where the feature vector is $d = [24, 25, 20, 22, 15, 17, 18, 16]^\top$, representing the ages of a group of people. A splitting candidate is chosen as Age = 18. If AP can infer that $m_i = [0, 0, 0, 0, 1, 1, 1, 1]$, it can conclude that the ages of the first four individuals are greater than those of the last four, leading to a privacy disclosure of the users' age data.

Due to the designed disturbing noise in Algorithm 1, AP is unable to deduce a potential solution for m_i through straightforward brute force enumeration. For each m_i , the output of MaskedXGBoost is the noise matrix B_i . Since this represents aggregated information about m_i , we employ differential privacy (DP) to assess the privacy leakage of PP.

THEOREM 3. *Recall that n is the number of instances in datasets, W is the number of noise vectors in a noise matrix B_i , i.e. $B_i = [b_{i1}, \dots, b_{iW}]$. For any ϵ_{PP} , δ_{PP} satisfy $\epsilon_{PP} > W \ln 2$, $\delta_{PP} \geq W \Pr_{y \sim \mathcal{N}(0, I_3)}(y^\top k y \geq 2\epsilon_{PP}/W - 2 \ln 2)$, MaskedXGBoost is $(\epsilon_{PP}, \delta_{PP})$ -differentially private for PP sending each noise matrix B_i , where $k = \text{diag}(k_1, k_2, k_3)$, k_1, k_2, k_3 are the eigenvalues of*

$$\begin{bmatrix} b-c & a+c & b-a \\ a-b & 2b & a-b \\ b-a & a+c & b-c \end{bmatrix}$$

$$\text{and } a = \frac{1}{n} \sum_{j=0}^{n-1} \frac{\sigma_1^2}{2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2}, \quad b = \frac{1}{n} \sum_{j=0}^{n-1} \frac{\sigma_1^2 \cos(\frac{2\pi j}{n})}{2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2} \text{ and } c = \frac{1}{n} \sum_{j=0}^{n-1} \frac{\sigma_1^2 \cos(\frac{4\pi j}{n})}{2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2}.$$

Table 2: Overview of the six datasets.

Dataset	n.	f.	p.	Dataset	n	f.	p.
Credit 1	150000	10	0.07	Higgs	200000	28	0.47
Adult	32651	14	0.24	Bank	45211	16	0.11
Credit 2	30000	23	0.22	Nomao	34465	10	0.28

We provide detailed proof of Theorem 3 in Appendix D. Based on Theorem 3, one can get a hint on the relation between noise ratio σ_1^2/σ_2^2 and privacy leakage in PP. According to our simulation results in Appendix D, with fixed ϵ_{PP} , the lower-bound for δ_{PP} increases monotonically with the noise ratio, indicating that a large value of σ_1^2/σ_2^2 translates to increased privacy leakage from PP. Intuitively, as the scale of disturbing noise σ_2 decreases, it becomes easier for AP to deduce the splitting vector by observing dimension additions close to zero, leveraging the insights from Algorithm 1.

6 EMPIRICAL EVALUATIONS

In this section, we first evaluate the model utility, convergence performance, and computational efficiency of MaskedXGBoost, respectively. Secondly, we conduct experiments to show the empirical privacy-preserving property of MaskedXGBoost. Thirdly, we conduct the ablation study to show how the PP privacy budget ϵ_{PP} affects MaskedXGBoost's performance.

6.1 Experimental Setup

Datasets. We conduct our experiments on six widely used public XGBoost datasets including Credit 1, Adult, Higgs, Bank, Credit 2, and Nomao. All datasets are displayed in Table 2 detailing the number of records(n .), number of features(f .), and proportion of the positive class (p .). All these real-world datasets are from Kaggle [26, 49] and the UCI repository [11].

Metrics. In the design of MaskedXGBoost, we mainly consider two aspects of performance, model utility, and computational efficiency.

- **(M1) Model Utility.** We use the AUC¹ score to measure the model utility, which is widely used to evaluate the prediction ability of XGBoost models on binary classification.
- **(M2) Computational Efficiency.** We calculate the *average training time per tree* to analyze the computational efficiency.

Competitors. We have four baselines: (1) XGBoost-NonPrivate [4]: Train XGBoost without privacy concerns. (2) XGBoost-LDP: Directly add Gaussian noise to gradients and Hessians to achieve LDP. (3) FedXGBoost-LDP [30]: Reduce the disturbance of LDP noises to the utility by modifying the splitting scoring function and utilizing only gradient information. (4) FederBoost [44]: Inject noise in data feature information of PP to achieve DP. The definition of PP’s privacy in this method is the same as in MaskedXGBoost, that is, the relative order of private feature values of training data samples.

Implementation. We implement MaskedXGBoost based on FedTree [33], which is an effective federated XGBoost template with C++. All experiments are run on a server containing 40 2.20 GHz CPUs, with 2 TB memory and Ubuntu 18.04 LTS system. All experiments are repeated 10 times and the mean and standard deviation are reported.

6.2 Utility Evaluation

In this section, we evaluate the model utility of different methods. The metric is the test AUC with 60 training rounds and tree depth $d = 6$. We choose ϵ_{AP} from $\{0.5, 1, 2, 4, 6, 8\}$. In all experiments we satisfy $(\epsilon_{AP}, \delta_{AP})$ -LDP and $(\epsilon_{PP}, \delta_{PP})$ -DP where $\delta_{AP} = 0.001$, and $\epsilon_{PP} = 1$, $\delta_{PP} = 1/n$. We use the advanced composition bound [15] to calculate the privacy budget for each step. XGBoost-NonPrivate represents the best AUC that can be achieved, and it is a horizontal line because its test AUC does not change with the privacy budget ϵ_{AP} . The evaluation results on four datasets are shown in Figure 4, and the other results are in Appendix H.

MaskedXGBoost’s Performance. As illustrated in Figure 4, we can clearly observe that MaskedXGBoost is far superior to the other two methods. With the increment of privacy budget ϵ_{AP} , the AUC of each method has been improved. MaskedXGBoost significantly outperforms the other baselines because a large portion of our noise used to protect privacy is carefully constructed and lossless.

Other Approaches’ Performance. In Figure 4, the AUCs of XGBoost-LDP and FedXGBoost-LDP have considerable fluctuations. During the XGBoost training process, if there is a significant error in the parent tree nodes, the error propagates to its left and right children nodes, resulting in larger errors. Consequently, the

¹Measures such as accuracy are not useful for testing the performance of models in cases of class imbalance.

Table 3: Efficiency evaluation of different methods on six datasets. The average training time per tree is reported. We highlight our method in the red ground.

Methods	Datasets					
	Credit 1	Adult	Higgs	Bank	Credit 2	Nomao
XGBoost-NonPrivate	0.25	0.15	0.58	0.17	0.38	0.15
MaskedXGBoost	3.95	1.54	11.17	2.21	5.17	1.77
HE-XGBoost	25.22	7.43	47.70	11.71	18.41	11.91
XGBoost-LDP	3.47	1.41	4.78	1.29	2.85	1.45
FedXGBoost-LDP	3.27	1.18	3.72	1.23	2.22	1.03
FederBoost	2.15	1.11	3.11	1.05	1.95	0.95

AUCs for these methods exhibit notable instability and do not show substantial enhancement even with increased ϵ_{AP} .

MaskedXGBoost sometimes performs on average similarly to FederBoost when the AP privacy budget is low (e.g., $\epsilon_{AP} = 0.5$ on Credit 1 and Adult), but it is more stable. Moreover, MaskedXGBoost significantly outperforms FederBoost as the AP privacy budget increases.

6.3 Convergence Evaluation

In order to better understand the differences between MaskedXGBoost and other approaches, we analyze the training process of each algorithm. Figure 5 depicts the training process of the four algorithms with different privacy budgets on Credit 1 dataset. The results on other datasets are in Appendix H.

MaskedXGBoost’s Performance. In general, we observe that the training process of the MaskedXGBoost is stable, and the test AUC gradually rises. As the privacy budget ϵ_{AP} grows, the AUC of MaskedXGBoost is getting higher and closer to the ground truth: XGBoost-NonPrivate. As the privacy budget is relaxed, the randomness of different runs gradually decreases, demonstrating the stability of our algorithm.

Other Approaches’ Performance. The training process of the other three algorithms has a large disturbance, even if the number of training rounds increases, the performance cannot be improved stably. FedXGBoost-LDP has less disturbance than XGBoost-LDP due to the adjustment of the scoring function. However, since FedXGBoost-LDP does not consider the Hessian information, its performance is degraded. The performance of XGBoost-LDP and FedXGBoost-LDP is not increased with the number of training rounds even with a relatively large privacy budget (i.e., $\epsilon_{AP} = 8$).

Since there is no AP privacy breach in FederBoost, its training process are not affected by the AP privacy budget. However, the FederBoost AUC results are highly unstable due to significant perturbations in the feature data, which disrupt XGBoost’s splitting candidate search process.

In XGBoost training, the error accumulates with the number of layers in a tree and also with the number of training rounds. Therefore, increasing the number of training rounds (number of trees) does not guarantee an improvement in performance in the presence of noise. MaskedXGBoost adaptively adjusts the noise scale during the training process and increases communication to

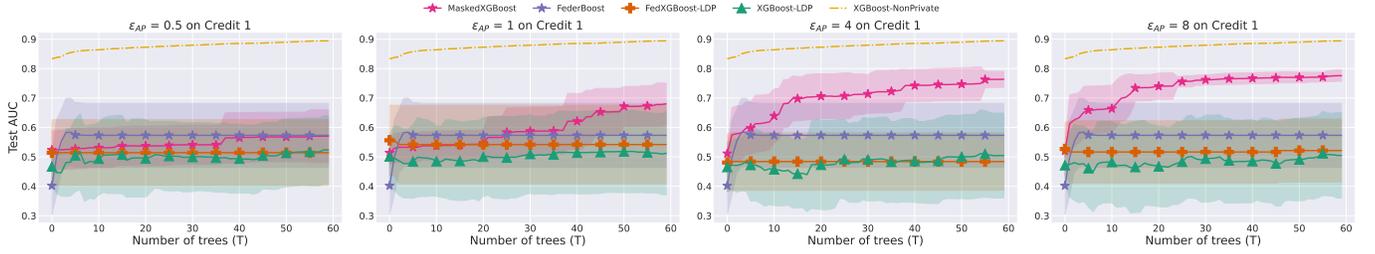


Figure 5: Training process of the four XGBoost algorithms with different privacy budgets on Credit 1 dataset

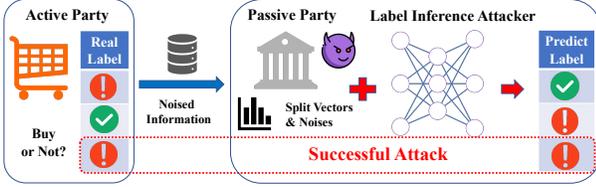


Figure 6: Illustration of the neural network-based attacker. PP receives the perturbed information and leverages the splitting vectors and noises in their possession to conduct a label inference attack using a pre-trained attacker model.

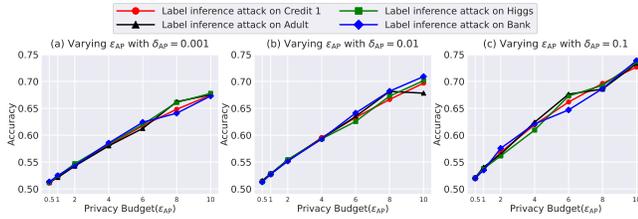


Figure 7: Empirical privacy evaluation for AP with varying privacy budget ϵ_{AP} and δ_{AP} . As long as the privacy budget is small enough, the attacker cannot infer successfully.

correct errors, resulting in reduced disturbance from noise during modeling.

6.4 Efficiency Evaluation

We show the training time comparison among MaskedXGBoost, XGBoost-NonPrivate, XGBoost-LDP, FedXGBoost-LDP, FederBoost, and HE-XGBoost (using the HE method Paillier to protect privacy). The computation overhead of our approach mainly comes from building the categorical matrix M and noise vectors. Thus, this overhead increases as the number of dimensions of training data increases. The majority of time consumption in HE-XGBoost arises from the encryption and decryption processes, as well as the ciphertext operations. Table 3 shows the average training time per tree of each method. The training time per tree for MaskedXGBoost is significantly lower than that of HE-XGBoost. MaskedXGBoost achieves 4.82x (Adult dataset) to 6.72x (Nomao dataset) training time improvement compared with the HE method. Although our algorithm is not as efficient as other DP algorithms, it is acceptable, and our utility is better. We provide the communication overhead analysis in Appendix F.

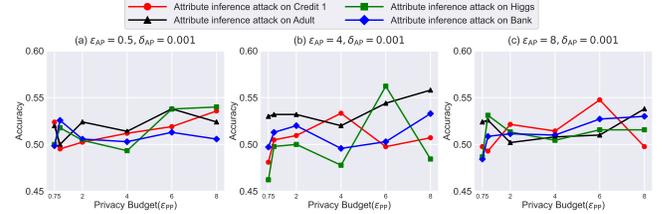


Figure 8: Empirical privacy evaluation for PP with varying privacy budget ϵ_{PP} and ϵ_{AP} . Even with a very loose privacy budget (e.g., $\epsilon_{PP} = 8, \delta_{AP} = 8$), attacks on PP’s splitting vector are still not effective.

6.5 Empirical Privacy Evaluation for AP

While we possess a theoretical privacy guarantee through local differential privacy, we also perform empirical privacy evaluations to confirm the adequacy of our algorithm’s privacy. In the honest-but-curious threat model, PP aims to perform a label inference attack. We introduce the following attacker for AP under this threat model.

Attack Setting & Goal. PP builds the noise vectors b_{i1}, \dots, b_{iW} according to a splitting vector m_i , and receive $\langle g \rangle_i, \langle h \rangle_i$ from AP, which are perturbed in virtue of (9). With the awareness of $b_{i1}, \dots, b_{iW}, \langle g \rangle_i, \langle h \rangle_i$, PP wants to de-noise $\langle g \rangle_i$ and $\langle h \rangle_i$. Then with the estimated plain information g and h , PP can infer the label of instances. Figure 6 shows the attack process of our attacker.

Attack Methodology. We construct a 5-layer neural network attacker for executing the attack. This neural network provides estimates for the plaintext information, denoted as g and h , which are subsequently used to calculate the predicted label. PP emulates the real communication process by introducing noise and applying it to the fabricated gradients and Hessians. These fabricated datasets are generated based on PP’s own splitting vector to simulate training data. For the testing, the attacker comprises the perturbed gradients and Hessians that AP sends during the actual execution of our method. We use L2 regularization with a parameter of 0.01 to mitigate overfitting.

Evaluation Results. After PP trains the attacker to convergence, we evaluate its performance by testing it on real, noisy gradients and Hessians, denoted as $\langle g \rangle_i$ and $\langle h \rangle_i$, which are generated by AP using our MaskedXGBoost. We conduct evaluations with different privacy budgets ϵ_{AP} and δ_{AP} on six datasets.

In Figure 7, we present the attack results on four datasets. Other results are available in Appendix H. Under a small privacy budget (e.g., $\epsilon_{AP} = 0.5, \delta_{AP} = 0.001$), the privacy of users can not be

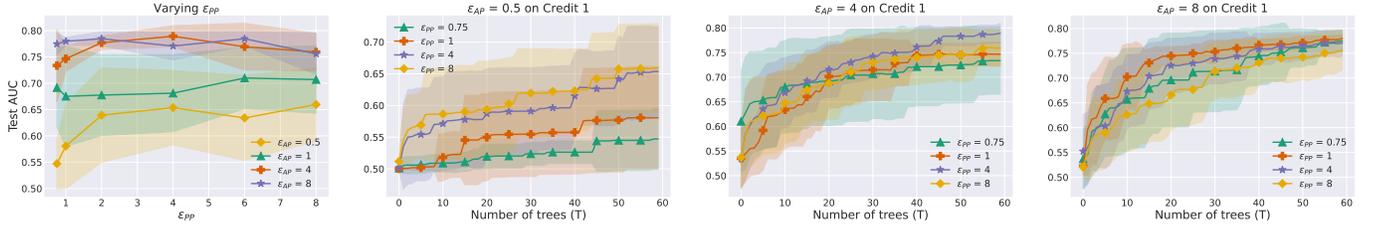


Figure 9: Ablation study on Credit 1: the effect of ϵ_{PP} on final test AUC and training process.

compromised. When the privacy budget parameters are relaxed, the effectiveness of attacks slightly improves.

6.6 Empirical Privacy Evaluation for PP

For the privacy of PP, besides providing theoretical differential privacy protection, we design an empirical privacy evaluation for PP. In the honest-but-curious threat model, AP aims to perform an attribute inference attack. Under this threat model, we introduce the following attacker for PP.

Attack Setting & Goal. AP receives the noise vectors b_{i1}, \dots, b_{iW} according to the splitting vector m_i owned by PP. Since the distribution of these noise vectors is related to the splitting vector m_i , the attacker wants to infer the private splitting vector m_i based on the noise vectors b_{i1}, \dots, b_{iW} .

Attack Methodology. Recall that noise consists of active noise, inactive noise, and disturbing noise in Section 4.2, where all dimensions of active noise add up to exactly 0. We assume that the attacker knows the size of the active set $n_{A,i}$. If no disturbing noise is injected, the attacker can traverse all possibilities $C_n^{n_{A,i}}$, where the one that adds exactly 0 is the correct active set. Due to the addition of disturbance noise, the corresponding dimensions of the active set are not added to 0, but the corresponding disturbance is still much smaller than that of other combinations. So the attacker can find the predicted active set based on the minimum absolute value of the sum.

Evaluation Results. We measure the attack effectiveness by the bit match between the splitting vector predicted by the attacker and the real vector. We conduct evaluations with different privacy budgets ϵ_{PP} and ϵ_{AP} and on six datasets. δ_{PP} and δ_{AP} is set to be $1/n, 0.001$ respectively.

In Figure 8, we present the attack results on four datasets. Other results are available in Appendix H. We find that the attack is ineffective due to the presence of disturbing noise, and even when the privacy budget is relaxed, the attack success rate does not exceed 0.6. This demonstrates that MaskedXGBoost fully guarantees the privacy of PP. The empirical privacy evaluation is based on the honest-but-curious adversary assumption, we provide the analysis if adversaries deviate from the protocol in Appendix G.

6.7 Ablation Study

Now we evaluate the impact of ϵ_{PP} on MaskedXGBoost’s performance. As shown in Figure 9, MaskedXGBoost’s utility improves with higher ϵ_{PP} values. This aligns with the findings in Theorem 1 and Theorem 3, where a larger ϵ_{PP} corresponds to a smaller σ_2 , allowing MaskedXGBoost to improve performance. Consider the training process, regardless of the ϵ_{PP} setting, MaskedXGBoost

exhibits a relatively fast convergence rate, although there may be a slight difference in the final test AUC. For other datasets’ results, please refer to Appendix H.

7 RELATED WORK

Vertical Federated XGBoost. Recently, Cheng et al. [6] proposed SecureBoost, a federated XGBoost framework for vertically partitioned data. In SecureBoost, AP calculates g_i and h_i for all samples, encrypts them using additively homomorphic encryption, and sends the ciphertexts to PP. Using the homomorphic property, PP computes the ciphertext sum of gradients and Hessians of left and right nodes for all possible splitting candidates and returns the ciphertexts to AP. AP decrypts all ciphertexts and finds the best split. This protocol is expensive since it requires cryptographic computation and communication for each possible split.

Some related works propose approaches that avoid encryption operations to get better efficiency. Tian et al. [44] suppose that PP adds differentially private noise to the users’ bucket, which is formed by the division of splitting candidates. Such an approach is still impractical because the user information in a bucket is highly compromised and performance decreases significantly as the privacy budget decreases.

Horizontal Federated XGBoost. Recent studies have explored GBDT implementations, such as XGBoost, for secure training in horizontal federated settings [8, 18]. These approaches typically rely on cryptographic techniques like Secure Multi-Party Computation (MPC) or HE, which result in high computational and communication overhead. Tian et al. [44] propose methods under the local DP model but experience significant utility loss due to the local noise [44]. Maddock et al. [37] use lightweight MPC techniques such as secure aggregation to balance DP efficiency with cryptographic security. Peinemann et al. [40] introduce non-spherical multivariate Gaussian noise to improve the utility-privacy trade-off.

8 CONCLUSION AND FUTURE WORK

In this paper, we proposed MaskedXGBoost that enables the state-of-the-art tree ensemble model XGBoost to be conducted under VFL settings. Different from the previous work applying homomorphic encryption, our protocol MaskedXGBoost achieves lower overhead while maintaining consistently better accuracy and provides a bilateral differential privacy guarantee with rigorous proof. In our protocol MaskedXGBoost, we design a special privacy protection mechanism for VFL XGBoost instead of directly applying DP noise perturbation, which may have great potential to be applied to other privacy-preserving algorithms.

In our future work, we plan to enhance the performance of MaskedXGBoost. This involves optimizing the construction of the categorical matrix and the matrix operation process. Additionally, we aim to implement MaskedXGBoost in real-world industrial scenarios to evaluate its performance with more realistic data.

REFERENCES

- [1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 308–318.
- [2] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security* 13, 5 (2017), 1333–1345.
- [3] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1175–1191.
- [4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*. 785–794.
- [5] Weijing Chen, Guoqiang Ma, Tao Fan, Yan Kang, Qian Xu, and Qiang Yang. 2021. Secureboost+: A high performance gradient boosting tree framework for large scale vertical federated learning. *arXiv preprint arXiv:2110.10927* (2021).
- [6] Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. 2021. Secureboost: A lossless federated learning framework. *IEEE Intelligent Systems* 36, 6 (2021), 87–98.
- [7] Philip J Davis. 1979. *Circulant Matrices* Wiley. Vol. 120. Wiley New York.
- [8] Kevin Deforth, Marc Desgroseilliers, Nicolas Gama, Mariya Georgieva, Dimitar Jetchev, and Marius Vuille. 2022. XORBoost: Tree boosting in the multiparty computation setting. *Proceedings on Privacy Enhancing Technologies* (2022).
- [9] Linkang Du, Zhikun Zhang, Shaohjie Bai, Changchang Liu, Shouling Ji, Peng Cheng, and Jiming Chen. 2021. AHEAD: Adaptive Hierarchical Decomposition for Range Query under Local Differential Privacy. In *ACM CCS*.
- [10] Yuntao Du, Yujia Hu, Zhikun Zhang, Ziquan Fang, Lu Chen, Baihua Zheng, and Yunjun Gao. 2023. LDPTrace: Locally Differentially Private Trajectory Synthesis. In *VLDB*.
- [11] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml>
- [12] Lutz Duembgen. 2010. Bounding standard gaussian tail probabilities. *arXiv preprint arXiv:1012.2063* (2010).
- [13] Cynthia Dwork, Krishnamurthy Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 486–503.
- [14] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.
- [15] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 51–60.
- [16] Ulfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1054–1067.
- [17] Wenjing Fang, Chaochao Chen, Jin Tan, Chaofan Yu, Yufei Lu, Li Wang, Lei Wang, Jun Zhou, and Alex X Liu. 2020. A hybrid-domain framework for secure gradient tree boosting. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*, Galway, Ireland. ACM, New York, NY, USA.
- [18] Zhi Feng, Haoyi Xiong, Chuanyuan Song, Sijia Yang, Baoxin Zhao, Licheng Wang, Zeyu Chen, Shengwen Yang, Liping Liu, and Jun Huan. 2019. Securegbm: Secure multi-party gradient boosting. In *2019 IEEE international conference on big data (big data)*. IEEE, 1312–1321.
- [19] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.
- [20] Fangcheng Fu, Jiawei Jiang, Yingxia Shao, and Bin Cui. 2019. An experimental evaluation of large scale GBDT systems. *arXiv preprint arXiv:1907.01882* (2019).
- [21] Nayanaba Pravinsinh Gohil and Arvind D Meniya. 2021. Click ad fraud detection using XGBoost gradient boosting algorithm. In *International Conference on Computing Science, Communication and Security*. Springer, 67–81.
- [22] Robert M Gray et al. 2006. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory* 2, 3 (2006), 155–239.
- [23] Roger A Horn and Charles R Johnson. 2012. *Matrix analysis*. Cambridge university press.
- [24] Lingying Huang, Junfeng Wu, Dawei Shi, Subhrakanti Dey, and Ling Shi. 2024. Differential privacy in distributed optimization with gradient tracking. *IEEE Trans. Automat. Control* (2024).
- [25] Chao Jin, Jun Wang, Sin G Teo, Le Zhang, CS Chan, Qibin Hou, and Khin Mi Mi Aung. 2022. Towards End-to-End Secure and Efficient Federated Learning for XGBoost. (2022).
- [26] Kaggle.2012. [n. d.]. Give Me Some Credit Competition Dataset. <https://www.kaggle.com/competitions/GiveMeSomeCredit/data?select=cs-test.csv>.
- [27] Alan F Karr, Xiaodong Lin, Ashish P Sanil, and Jerome P Reiter. 2009. Privacy-preserving analysis of vertically partitioned data using secure matrix products. *Journal of Official Statistics* 25, 1 (2009), 125.
- [28] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems* 30 (2017).
- [29] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. 2017. Practical multi-party private set intersection from symmetric-key techniques. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1257–1272.
- [30] Nhan Khanh Le, Yang Liu, Quang Minh Nguyen, Qingchen Liu, Fangzhou Liu, Quanwei Cai, and Sandra Hirche. 2021. FedXGBoost: Privacy-Preserving XGBoost for Federated Learning. *arXiv preprint arXiv:2106.10662* (2021).
- [31] Jerome Le Ny and George J Pappas. 2013. Differentially private filtering. *IEEE Trans. Automat. Control* 59, 2 (2013), 341–354.
- [32] Oscar Li, Jiankai Sun, Xin Yang, Weihaio Gao, Hongyi Zhang, Junyuan Xie, Virginia Smith, and Chong Wang. 2021. Label Leakage and Protection in Two-party Split Learning. In *International Conference on Learning Representations*.
- [33] Qinbin Li, Zhaomin Wu, Yanzheng Cai, Yuxuan Han, Ching Man Yung, Tianyuan Fu, and Bingsheng He. 2023. FedTree: A Federated Learning System For Trees. In *Proceedings of Machine Learning and Systems*.
- [34] Qinbin Li, Zhaomin Wu, Zeyi Wen, and Bingsheng He. 2020. Privacy-preserving gradient boosting decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 784–791.
- [35] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model ensemble for click prediction in bing search ads. In *Proceedings of the 26th international conference on world wide web companion*. 689–698.
- [36] Wen-jie Lu, Zhicong Huang, Qizhi Zhang, Yuchen Wang, and Cheng Hong. 2023. Squirrel: A Scalable Secure {Two-Party} Computation Framework for Training Gradient Boosting Decision Tree. In *32nd USENIX Security Symposium (USENIX Security 23)*. 6435–6451.
- [37] Samuel Maddock, Graham Cormode, Tianhao Wang, Carsten Maple, and Somesh Jha. 2022. Federated Boosted Decision Trees with Differential Privacy. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2249–2263.
- [38] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*, Aarti Singh and Jerry Zhu (Eds.). PMLR, 1273–1282. <https://proceedings.mlr.press/v54/mcmahan17a.html>
- [39] H Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated learning of deep networks using model averaging. *arXiv preprint arXiv:1602.05629* 2 (2016).
- [40] Thorsten Peinemann, Moritz Kirschte, Joshua Stock, Carlos Cotrini, and Esfandiar Mohammadi. 2023. S-BDT: Distributed Differentially Private Boosted Decision Trees. *arXiv e-prints* (2023), arXiv–2309.
- [41] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1310–1321.
- [42] Garry J Tee. 2007. Eigenvectors of block circulant and alternating circulant matrices. *New Zealand Journal of Mathematics* 36, 8 (2007), 195–211.
- [43] Zhenya Tian, Jialiang Xiao, Haonan Feng, and Yutian Wei. 2020. Credit risk assessment based on gradient boosting decision tree. *Procedia Computer Science* 174 (2020), 150–160.
- [44] Zhihua Tian, Rui Zhang, Xiaoyang Hou, Lingjuan Lyu, Tianyi Zhang, Jian Liu, and Kui Ren. 2023. FederBoost: Private Federated Learning for GBDT. *IEEE Transactions on Dependable and Secure Computing* (2023).
- [45] Haiming Wang, Zhikun Zhang, Tianhao Wang, Shibo He, Michael Backes, Jiming Chen, and Yang Zhang. 2023. PrivTrace: Differentially Private Trajectory Synthesis by Adaptive Markov Model. In *USENIX Security*.
- [46] Rui Wang, Oguzhan Ersoy, Hangyu Zhu, Yaochu Jin, and Kaitai Liang. 2022. Feverless: Fast and secure vertical federated learning based on xgboost for decentralized labels. *IEEE Transactions on Big Data* (2022).
- [47] Tianhao Wang, Joann Qiongna Chen, Zhikun Zhang, Dong Su, Yueqiang Cheng, Zhou Li, Ninghui Li, and Somesh Jha. 2021. Continuous Release of Data Streams under both Centralized and Local Differential Privacy. In *ACM CCS*.
- [48] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, 2 (2019), 1–19.

- [49] I-Cheng Yeh and Che-hui Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications* 36, 2 (2009), 2473–2480.
- [50] Quan Yuan, Zhikun Zhang, Linkang Du, Min Chen, Peng Cheng, and Mingyang Sun. 2023. PrivGraph: Differentially Private Graph Data Publication by Exploiting Community Information. In *USENIX Security*.
- [51] Bokang Zhang, Yanglin Zhang, Zhikun Zhang, Jinglan Yang, Lingying Huang, and Junfeng Wu. 2024. S2NeRF: Privacy-preserving Training Framework for NeRF. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 258–272.
- [52] Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, and Jiming Chen. 2018. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy. In *ACM CCS*.
- [53] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. 2021. PrivSyn: Differentially Private Data Synthesis. In *USENIX Security*.

A DETAILS OF XGBOOST

Regression Tree. A regression tree, which belongs to the family of decision trees, is a commonly used algorithm for predicting numerical output variables in a given dataset, where the numerical output variables are defined as

$$f(x) = w_{q(x)}, q: X \rightarrow \{1, \dots, L\}, w \in \mathbb{R}^L$$

where q denotes the tree structure that maps the feature of an instance to a unique leaf, w is the weight vector of the leaf, and L is the number of leaves of one tree.

The regression tree is capable of assigning any instance x to a specific leaf node with a leaf index $q(x)$. This index corresponds to a unique leaf weight value $w_{q(x)}$. The process of training a regression tree is essentially learning a *tree structure* q and *leaf weight* w .

Splitting Rules. To learn a tree structure at the t^{th} iteration, many *splitting rules*, i.e., mappings that assign an instance x_i to a specific leaf node, are proposed according to the order of feature data [19]. The algorithm proposed in [4] is a greedy rule that splits the instances into two disjoint sets of instances (which are associated with the left and right child nodes) at a node and repeats the splitting multiple times. (6) is often used to evaluate a splitting candidate at each node j (associated with instance set \mathcal{I}_j) [4]. In (6), $\mathcal{I}_{L,j}$ and $\mathcal{I}_{R,j}$ are associated with the left and right child nodes, forming the dichotomy of the proposed splitting in question, and λ and γ are the regularization parameters in (2).

The proposed that gets the highest score in (6) will be selected as the tree structure at this node. Then, splitting continues from the newly constructed nodes until it reaches the maximum depth of the tree. The deepest nodes are called the leaves of a tree.

B PROOF OF THEOREM 1

We use L_{split}^{si} to denote the true splitting score of the i^{th} splitting candidate and $\langle L_{split}^{si} \rangle$ to denote MaskedXGBoost's estimation of it. m_i is the corresponding splitting vector. $\langle g \rangle_i, \langle h \rangle_i \in \mathbb{R}^n$ are the noised gradient and Hessian vectors of the n instances. G and H are the sums of all gradients and Hessians. We have

$$L_{split}^{si} = -\gamma + \frac{1}{2} \left(\frac{(g_i^L)^2}{(h_i^L) + \lambda} + \frac{(g_i^R)^2}{(h_i^R) + \lambda} - \frac{G^2}{H + \lambda} \right).$$

$$\langle L_{split}^{si} \rangle = -\gamma + \frac{1}{2} \left(\frac{(\langle g_i^L \rangle)^2}{\langle h_i^L \rangle + \lambda} + \frac{(\langle g_i^R \rangle)^2}{\langle h_i^R \rangle + \lambda} - \frac{G^2}{H + \lambda} \right).$$

where $\langle g_i^L \rangle = m_i^\top \langle g \rangle_i, \langle h_i^L \rangle = m_i^\top \langle h \rangle_i, \langle g_i^R \rangle = G - \langle g_i^L \rangle, \langle h_i^R \rangle = H - \langle h_i^L \rangle$. According to Section 4.2, we have $\mathbb{E}[\langle g_i^L \rangle] = g_i^L, \text{Var}[\langle g_i^L \rangle] = \text{Var}[\langle g_i^R \rangle] = n_A C \sigma_2^2$, and we denote the variance by κ^2 in Theorem 1.

We want to upper bound:

$$\Pr \left(| \langle L_{split}^{si} \rangle - L_{split}^{si} | \geq \alpha \right) \quad \forall \alpha > 0 \quad (14)$$

Denote $X := \langle g_i^L \rangle, Y := \langle h_i^L \rangle + \lambda, Z := \langle g_i^R \rangle$ and $W := \langle h_i^R \rangle + \lambda$, then X, Y, Z, W are normal with variance κ^2 and mean $\mu_X, \mu_Y, \mu_Z, \mu_W$. Denote $C_L := \frac{(g_i^L)^2}{(h_i^L) + \lambda}$ and $C_R := \frac{(g_i^R)^2}{(h_i^R) + \lambda}$. We can write (14) as:

$$\Pr \left(\left| \frac{X^2}{Y} - C_L + \frac{Z^2}{W} - C_R \right| \geq 2\alpha \right). \quad (15)$$

We have (15) upper bounded by $\Pr(L) + \Pr(R)$, as short hands for $\Pr \left(\left| \frac{X^2}{Y} - C_L \right| \geq \alpha \right)$ and $\Pr \left(\left| \frac{Z^2}{W} - C_R \right| \geq \alpha \right)$, respectively. It's also possible to derive a tighter bound $\Pr(L) + \Pr(R) - \Pr(L \cap R)$, but much more tedious.

For simplicity, we only present the derivation of $\Pr(L)$ while omit the derivation for $\Pr(R)$, which is very similar. Denote $C_L + \alpha := \beta_L, C_L - \alpha := \beta'_L$, we proceed by conditioning L on the sign of Y :

$$\Pr(L) = \Pr(A \cap \{Y > 0\}) + \Pr(B \cap \{Y > 0\}) \\ + \Pr(B^c \cap \{Y < 0\}) + \Pr(A^c \cap \{Y < 0\}) \quad (16)$$

where $A := \{X^2 \geq \beta_L Y\}, B := \{X^2 \leq \beta'_L Y\}$.

We can write the first term in (16) as an integration:

$$\int_0^\infty \Pr(X^2 \geq \beta_L t) dF_Y(t) \\ = \int_0^\infty \left(\Phi \left(\frac{-\sqrt{\beta_L t} + \mu_X}{\kappa} \right) + \Phi \left(\frac{-\sqrt{\beta_L t} - \mu_X}{\kappa} \right) \right) dF_Y(t)$$

where $dF_Y(t) = \frac{1}{\kappa} \phi \left(\frac{t - \mu_Y}{\kappa} \right) dt$, $\phi(\cdot)$ and $\Phi(\cdot)$ are the density and cdf of the standard normal distribution.

Similar integration can be done for the second and third terms in (16), while the last term is zero. In summary:

- If $\alpha \leq C_L$, i.e. $\beta'_L \geq 0$, $\Pr(L)$ is no greater than:

$$\Pr(A \cap \{Y > 0\}) + \Pr(B \cap \{Y > 0\}) + \Pr(\{Y < 0\}) \quad (17)$$

- if $\alpha \geq C_L$, i.e. $\beta'_L \leq 0$, $\Pr(L)$ is no greater than:

$$\Pr(A \cap \{Y > 0\}) + \Pr(B^c \cap \{Y < 0\}) \quad (18)$$

As $\kappa^2 \downarrow 0, Y \xrightarrow{a.s.} \mu_Y, X^2 \xrightarrow{a.s.} \mu_X^2$. From the facts that $C_L \cdot \mu_Y = \mu_X^2$ and $\mu_Y > 0$, we conclude that as $\kappa^2 \downarrow 0, \Pr(A), \Pr(B)$, and $\Pr(\{Y < 0\})$ converge to zero; as a result, the upper bounds (17) and (18) converge to zero. ■

REMARK 3. To facilitate numerical evaluation, we can further use various inequalities [12] for $\Phi(\cdot)$ in integration, such as:

$$\frac{2\phi(x)}{\sqrt{4+x^2+x}} < 1 - \Phi(x) < \frac{2\phi(x)}{\sqrt{8/\pi+x^2+x}} \quad \forall x > 0$$

C PROOF OF THEOREM 2

Assume $g, g' \in \mathbb{R}^n$ are any two gradient vectors of n instances from AP, $g = [g_1, \dots, g_n]^\top, g' = [g'_1, \dots, g'_n]^\top$. Let $v = g - g' = [v_1, \dots, v_n]$, and we have $|v_i| \leq \mu$. For each splitting vector m_i , our algorithm is equivalent to adding a noise vector on g : $\langle g \rangle_i = g + \eta_i, \eta_i \sim \mathcal{N}(0, \Sigma)$, where the submatrix with rows and columns indexed by $\mathcal{A}(m_i)$ is given as

$$\Sigma_2 = \begin{bmatrix} C(2\sigma_1^2 + \sigma_2^2) & -C\sigma_1^2 & 0 & \cdots & 0 & -C\sigma_1^2 \\ -C\sigma_1^2 & \ddots & \ddots & \ddots & & 0 \\ 0 & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & & & & \ddots & -C\sigma_1^2 \\ -C\sigma_1^2 & 0 & \cdots & 0 & -C\sigma_1^2 & C(2\sigma_1^2 + \sigma_2^2) \end{bmatrix} \quad (19)$$

and the submatrix of all the other entries is given as

$$\Sigma_1 = C(2\sigma_1^2 + \sigma_2^2)I. \quad (20)$$

Notice that $\Sigma_1 \in \mathbb{R}^{n_{I,i} \times n_{I,i}}, \Sigma_2 \in \mathbb{R}^{n_{A,i} \times n_{A,i}}$, where $n_{I,i} = |I(m_i)|, n_{A,i} = |\mathcal{A}(m_i)|$. Without loss of generality, we assume $\Sigma = \text{diag}(\Sigma_1, \Sigma_2)$. For any measurable set $\mathcal{O} \subset \mathbb{R}^n$, we have

$$\begin{aligned} & \Pr(\langle g \rangle_i \in \mathcal{O}) \\ &= \int_{\mathbb{R}^n} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} 1_{\mathcal{O}}(g + w) e^{-\frac{1}{2} w^\top \Sigma^{-1} w} dw \\ &= \int_{\mathbb{R}^n} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} 1_{\mathcal{O}}(u) e^{-\frac{1}{2} (u-g)^\top \Sigma^{-1} (u-g)} du \\ &= \int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2} (u-g')^\top \Sigma^{-1} (u-g')} e^{(u-g')^\top \Sigma^{-1} v - \frac{1}{2} v^\top \Sigma^{-1} v} du \\ &\leq \int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2} (u-g)^\top \Sigma^{-1} (u-g)} 1_{\mathcal{S}}(u) du \\ &\quad + \exp(\varepsilon_{\text{AP}}) \Pr(\langle g' \rangle_i \in \mathcal{O}), \end{aligned}$$

where $\mathcal{S} := \{u \in \mathbb{R}^n | e^{(u-g')^\top \Sigma^{-1} v - \frac{1}{2} v^\top \Sigma^{-1} v} \geq e^{\varepsilon_{\text{AP}}}\}$. Now we proceed to bound the integral term.

$$\begin{aligned} & \int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2} (u-g)^\top \Sigma^{-1} (u-g)} 1_{\mathcal{S}}(u) du \\ &\leq \int_{\mathbb{R}^n} \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2} (u-g)^\top \Sigma^{-1} (u-g)} 1_{\mathcal{S}}(u) du \\ &= \Pr((u-g')^\top \Sigma^{-1} v - \frac{1}{2} v^\top \Sigma^{-1} v \geq \varepsilon_{\text{AP}}) \\ &= \Pr_{w \sim \mathcal{N}(0, \Sigma)}((w+v)^\top \Sigma^{-1} v - \frac{1}{2} v^\top \Sigma^{-1} v \geq \varepsilon_{\text{AP}}) \\ &= \Pr_{w \sim \mathcal{N}(0, \Sigma)}(w^\top \Sigma^{-1} v + \frac{1}{2} v^\top \Sigma^{-1} v \geq \varepsilon_{\text{AP}}) \end{aligned}$$

We denote $v = [v_1^\top, v_2^\top]^\top, v_1 \in \mathbb{R}^{n_{I,i}}, v_2 \in \mathbb{R}^{n_{A,i}}$, then we bound the above term as follows:

$$\begin{aligned} & \Pr(y^\top \Sigma^{-\frac{1}{2}} v + \frac{1}{2} (v_1^\top \Sigma_1^{-1} v_1 + v_2^\top \Sigma_2^{-1} v_2) \geq \varepsilon_{\text{AP}}) \\ &\leq \Pr(y^\top \Sigma^{-\frac{1}{2}} v \geq \varepsilon_{\text{AP}} - \frac{n_{I,i} \mu^2}{2C(2\sigma_1^2 + \sigma_2^2)} - \frac{n_{A,i} \mu^2}{2C\sigma_2^2}) \end{aligned}$$

where $y \sim \mathcal{N}(0, I_n), I_n$ is the n -by- n identity matrix. The inequality follows from the eigenvalues bound of Σ_2^{-1} [42]:

$$\lambda_j = \frac{1}{C(2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2)} \leq \frac{1}{C\sigma_2^2}$$

Notice that $y^\top \Sigma^{-\frac{1}{2}} v \sim \mathcal{N}(0, \|\Sigma^{-\frac{1}{2}} v\|_2^2)$, and

$$\|\Sigma^{-\frac{1}{2}} v\|_2^2 = v^\top \Sigma^{-1} v \leq \frac{n_{I,i} \mu^2}{C(2\sigma_1^2 + \sigma_2^2)} + \frac{n_{A,i} \mu^2}{C\sigma_2^2}$$

Let $\sigma^2 = \frac{n_{I,i} \mu^2}{C(2\sigma_1^2 + \sigma_2^2)} + \frac{n_{A,i} \mu^2}{C\sigma_2^2}$, we have

$$\Pr(y^\top \Sigma^{-\frac{1}{2}} v \geq \varepsilon_{\text{AP}} - \frac{\sigma^2}{2}) \leq \Pr_{z \sim \mathcal{N}(0, \sigma^2)}(z \geq \varepsilon_{\text{AP}} - \frac{\sigma^2}{2})$$

If $\sigma^2 \leq 2(\varepsilon_{\text{AP}} - 2 \ln \delta_{\text{AP}} - 2\sqrt{\ln \delta_{\text{AP}} (\ln \delta_{\text{AP}} - \varepsilon_{\text{AP}})})$, we can verify that

$$\Pr_{z \sim \mathcal{N}(0, \sigma^2)}(z \geq \frac{\sigma^2}{2}) \leq \delta_{\text{AP}}$$

where the last inequality follows from a Gaussian tail bound.

D PROOF OF THEOREM 3

Without loss of generality, we consider two adjacent splitting vectors m_i, m'_i that only differ in the first entry (i.e., $m_{i1} = 0, m'_{i1} = 1$) and with all other entries being 1. The other cases can be proved similarly. With m_i and m'_i , PP generates two noise matrices, denoted as B_i and B'_i . Each column vector $b_{ij} (j = 1, \dots, W)$ of the matrix B_i is randomly and independently drawn from $\mathcal{N}(0, \Sigma_1)$ and each column vector $b'_{ij} (j = 1, \dots, W)$ of the matrix B'_i randomly and independently drawn from $\mathcal{N}(0, \Sigma_2)$, where Σ_1 and Σ_2 are shown as follows:

$$\Sigma_2 = \begin{bmatrix} 2\sigma_1^2 + \sigma_2^2 & -\sigma_1^2 & 0 & \cdots & 0 & -\sigma_1^2 \\ -\sigma_1^2 & \ddots & \ddots & \ddots & & 0 \\ 0 & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & & & & \ddots & -\sigma_1^2 \\ -\sigma_1^2 & 0 & \cdots & 0 & -\sigma_1^2 & 2\sigma_1^2 + \sigma_2^2 \end{bmatrix}$$

$$\Sigma_1 = \begin{bmatrix} -\frac{2\sigma_1^2 + \sigma_2^2}{\sigma_1^2} & \vdots \\ 0 & -\frac{2\sigma_1^2 + \sigma_2^2}{\sigma_1^2} & -\frac{0}{\sigma_1^2} & -\frac{0}{\sigma_1^2} & -\frac{0}{\sigma_1^2} & \cdots & -\frac{0}{\sigma_1^2} & -\frac{0}{\sigma_1^2} & -\frac{0}{\sigma_1^2} & -\frac{0}{\sigma_1^2} \\ 0 & \vdots & -\sigma_1^2 & \ddots & \ddots & \ddots & & & & 0 \\ 0 & \vdots & 0 & \ddots & \ddots & \ddots & & & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \ddots & & & & 0 \\ 0 & \vdots & 0 & & & \ddots & & & & -\sigma_1^2 \\ 0 & \vdots & -\sigma_1^2 & 0 & \cdots & 0 & -\sigma_1^2 & 2\sigma_1^2 + \sigma_2^2 & & \end{bmatrix}$$

Since each column of B_i is generated independently, we can only consider b_{i1} and b'_{i1} and apply Lemma 1 to obtain the result for the whole matrix.

Denote $\Sigma := \Sigma_1^{-1} - \Sigma_2^{-1}$ and $\mathcal{S} := \{w : \sqrt{\frac{|\Sigma_2|}{|\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma w} \geq e^\varepsilon\}$, for any measurable set $\mathcal{O} \subset \mathbb{R}^n$, we have

$$\begin{aligned} \Pr(b_{i1} \in \mathcal{O}) &= \int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma_1^{-1} w} dw \\ &= \int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma_2^{-1} w} e^{-\frac{1}{2} w^\top \Sigma w} dw \\ &= \int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma_2|}} e^{-\frac{1}{2} w^\top \Sigma_2^{-1} w} \sqrt{\frac{|\Sigma_2|}{|\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma w} dw \\ &\leq \int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma_1^{-1} w} \mathbf{1}_{\mathcal{S}}(w) dw \\ &\quad + \exp(\varepsilon) \Pr(b'_{i1} \in \mathcal{O}), \end{aligned}$$

Now we proceed to bound the integral term. The following fact is used in the derivation: the determinant ratio $\frac{|\Sigma_2|}{|\Sigma_1|} \in (\frac{1}{4}, 2)$, and we provide the proof of it in Appendix E.

$$\begin{aligned} &\int_{\mathcal{O}} \frac{1}{\sqrt{(2\pi)^n |\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma_1^{-1} w} \mathbf{1}_{\mathcal{S}}(w) dw \\ &\leq \int_{\mathbb{R}^n} \frac{1}{\sqrt{(2\pi)^n |\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma_1^{-1} w} \mathbf{1}_{\mathcal{S}}(w) dw \\ &= \Pr_{w \sim \mathcal{N}(0, \Sigma_1)} \left(\sqrt{\frac{|\Sigma_2|}{|\Sigma_1|}} e^{-\frac{1}{2} w^\top \Sigma w} \geq e^\varepsilon \right) \\ &< \Pr_{w \sim \mathcal{N}(0, \Sigma_1)} \left(2e^{-\frac{1}{2} w^\top \Sigma w} \geq e^\varepsilon \right) \\ &= \Pr_{w \sim \mathcal{N}(0, \Sigma_1)} \left(w^\top (-\Sigma) w \geq 2\varepsilon - 2 \ln 2 \right) \\ &= \Pr_{y \sim \mathcal{N}(0, I)} \left(y^\top \Sigma_1^{\frac{1}{2}} (\Sigma_2^{-1} - \Sigma_1^{-1}) \Sigma_1^{\frac{1}{2}} y \geq 2\varepsilon - 2 \ln 2 \right) \\ &= \Pr_{y \sim \mathcal{N}(0, I)} \left(y^\top (\Sigma_1^{\frac{1}{2}} \Sigma_2^{-1} \Sigma_1^{\frac{1}{2}} - I) y \geq 2\varepsilon - 2 \ln 2 \right) \end{aligned}$$

Because $\Sigma_1^{\frac{1}{2}} \Sigma_2^{-1} \Sigma_1^{\frac{1}{2}} - I$ is symmetric, we have unitary decomposition

$$\Sigma_1^{\frac{1}{2}} \Sigma_2^{-1} \Sigma_1^{\frac{1}{2}} - I = U \Lambda U^\top$$

where U is orthogonal, and Λ is a diagonal matrix composed of the eigenvalues of $\Sigma_1^{\frac{1}{2}} \Sigma_2^{-1} \Sigma_1^{\frac{1}{2}} - I$, where $y \sim \mathcal{N}(0, I)$, we have $U^\top y \sim \mathcal{N}(0, I)$. It follows that

$$\begin{aligned} &\Pr_{y \sim \mathcal{N}(0, I)} \left(y^\top (\Sigma_1^{\frac{1}{2}} \Sigma_2^{-1} \Sigma_1^{\frac{1}{2}} - I) y \geq 2\varepsilon - \ln 2 \right) \\ &= \Pr_{y \sim \mathcal{N}(0, I)} \left(y^\top \Lambda y \geq 2\varepsilon - 2 \ln 2 \right) \end{aligned}$$

Next, we prove that Λ only has three non-zero diagonals. By Theorem 1.3.22 of [23], the set of the eigenvalues of $\Sigma_1^{\frac{1}{2}} \Sigma_2^{-1} \Sigma_1^{\frac{1}{2}} - I$ are equal to that of $\Sigma_2^{-1} \Sigma_1 - I$, which are further equal to that of $(P \Sigma_2 P^\top)^{-1} P (\Sigma_1 - \Sigma_2) P^\top$, where P can be any orthogonal matrix. Specifically, we choose the following permutation matrix:

$$P = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

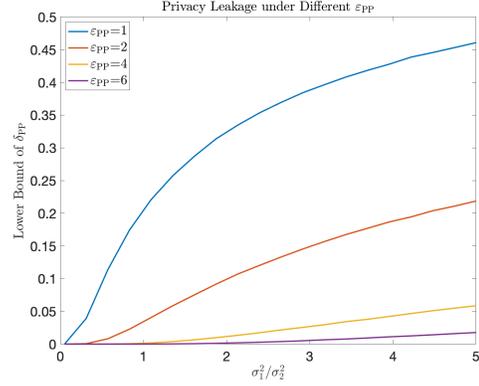


Figure 10: Simulation results for Theorem 3: Greater σ_1^2/σ_2^2 values correlate with increased privacy leakage.

Notice that

$$P \Sigma_2 P^\top = \Sigma_2$$

$$P (\Sigma_1 - \Sigma_2) P^\top = \sigma_1^2 \begin{bmatrix} 0 & 1 & -1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & \cdots & 0 \\ -1 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}_{n \times n} \triangleq \Delta$$

As a result, the set of the eigenvalues of $\Sigma_1^{\frac{1}{2}} \Sigma_2^{-1} \Sigma_1^{\frac{1}{2}} - I$ is equivalent to that of $\Sigma_2^{-1} \Delta$.

Noticed that only the first three columns of $\Sigma_2^{-1} \Delta$ have non-zero entries. Therefore, $\Sigma_2^{-1} \Delta$ only has three non-zero real eigenvalues which are also the eigenvalues of the upper left 3×3 block matrix, denoted as $[\Sigma_2^{-1} \Delta]_{3 \times 3}$.

As Σ_2 is a symmetric circulant matrix, the eigenvalues $\{\lambda_j, j = 0 \dots n-1\}$ and eigenvectors $\{\mathbf{u}_j, j = 0 \dots n-1\}$ of Σ_2^{-1} write as follows [22]:

$$\lambda_j = \frac{1}{2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2}$$

$$\mathbf{u}_j = \frac{1}{\sqrt{n}} \left[1, \cos(\frac{2\pi j}{n}), \cos(\frac{4\pi j}{n}), \dots, \cos(\frac{(n-1)2\pi j}{n}) \right]^\top$$

Let $r := \frac{\sigma_2^2}{2\sigma_1^2}$. Use the above formula, we write $[\Sigma_2^{-1} \Delta]_{3 \times 3}$ explicitly. Denote

$$\begin{aligned} a(r) &\triangleq \frac{1}{n} \sum_{j=0}^{n-1} \frac{\sigma_1^2}{2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2} \\ b(r) &\triangleq \frac{1}{n} \sum_{j=0}^{n-1} \frac{\sigma_1^2 \cos(\frac{2\pi j}{n})}{2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2} \\ c(r) &\triangleq \frac{1}{n} \sum_{j=0}^{n-1} \frac{\sigma_1^2 \cos(\frac{4\pi j}{n})}{2\sigma_1^2 [1 - \cos(\frac{2\pi j}{n})] + \sigma_2^2} \end{aligned}$$

We have

$$[\Sigma_2^{-1}\Delta]_{3 \times 3} = \begin{bmatrix} b-c & a+c & b-a \\ a-b & 2b & a-b \\ b-a & a+c & b-c \end{bmatrix}$$

Notice that $k_2(r) = a - c$ is an eigenvalue with eigenvector $\frac{1}{\sqrt{2}}[1, 0, 1]^\top$, and the other two satisfy

$$k_1(r) + k_3(r) = 4b - a - c \quad k_1(r)k_3(r) = 2(b^2 - a^2 + b^2 - ac)$$

We have following bound: for any measurable set $O \subset \mathbb{R}^n$,

$$\Pr(b_{i1} \in O) \leq \delta(\varepsilon, r) + \exp(\varepsilon)\Pr(b'_{i1} \in O)$$

where

$$\delta(\varepsilon, r) = \Pr_{y \sim \mathcal{N}(0, I)}(k_1(r)y_1^2 + k_2(r)y_2^2 + k_3(r)y_3^2 \geq 2\varepsilon - \ln 2)$$

Since y_1, y_2, y_3 are independent, $\delta(\varepsilon, r)$ can be easily obtained by a Monte Carlo simulation. Similarly, we can show that $O \subset \mathbb{R}^n$,

$$\Pr(b'_{i1} \in O) \leq \delta(\varepsilon, r) + \exp(\varepsilon)\Pr(b_{i1} \in O)$$

Finally, we apply Lemma 1 to obtain the $(\varepsilon_{pp}, \delta_{pp})$ -DP result for Algorithm 3, and $\varepsilon_{pp} = W\varepsilon$, $\delta_{pp} = W\delta(\varepsilon, r)$.

We also include numerical simulations with $W = 1$ in Figure 10 to confirm our intuition that larger values of σ_1^2/σ_2^2 result in increased privacy leakage. When σ_1^2/σ_2^2 is raised, holding ε_{pp} constant, we observe a corresponding increase in the lower bound of δ_{pp} , signifying heightened privacy leakage. ■

E PROOF OF THE DETERMINANT BOUND

By [22, Theorem3.1], we have $\frac{|\Sigma_2|}{|\Sigma_1|} = f(r, n)$ where

$$\begin{aligned} \ln f(r, n) &= -\ln(1+r) + \sum_{i=0}^n \ln\left(1+r - \cos\left(\frac{2\pi i}{n}\right)\right) \\ &\quad - \sum_{i=0}^{n-1} \ln\left(1+r - \cos\left(\frac{2\pi i}{n-1}\right)\right) \end{aligned}$$

E.1 Upper Bound

Firstly, we can assume n is even. Denote

$$T(n, i) := \ln\left(1+r - \cos\left(\frac{2\pi i}{n}\right)\right)$$

One has

$$\begin{aligned} \ln f(r, n) &= -\ln(1+r) + \sum_{i=0}^n T(n, i) - \sum_{i=0}^{n-1} T(n-1, i) \\ &= \ln \frac{2+r}{1+r} + 2 \sum_{i=0}^{\frac{n}{2}-1} (T(n, i) - T(n-1, i)) \\ &< \ln \frac{2+r}{1+r} \end{aligned}$$

where the last inequality comes from

$$\begin{aligned} \cos\left(\frac{2\pi i}{n}\right) &> \cos\left(\frac{2\pi i}{n-1}\right), i = 1, \dots, \frac{n}{2} - 1; \\ \cos\left(\frac{2\pi i}{n}\right) &> \cos\left(\frac{2\pi(i-1)}{n-1}\right), i = \frac{n}{2} + 1, \dots, n-1 \end{aligned}$$

Thus, when n is even, one has $f(r, n) < 2$. A similar proof works for the case where n is odd.

E.2 Lower Bound

We define

$$\begin{aligned} \ln g(r, n) &= \ln f(r, n) - \ln(2+r) + \ln(1+r) \\ &= -\ln(2+r) + \sum_{i=0}^n \ln\left(1+r - \cos\left(\frac{2\pi i}{n}\right)\right) - \sum_{i=0}^{n-1} \ln\left(1+r - \cos\left(\frac{2\pi i}{n-1}\right)\right) \end{aligned}$$

We have

$$\frac{\partial \ln g(r, n)}{\partial r} = -\frac{1}{2+r} + \sum_{i=0}^n \frac{1}{1+r - \cos\left(\frac{2\pi i}{n}\right)} - \sum_{i=0}^{n-1} \frac{1}{1+r - \cos\left(\frac{2\pi i}{n-1}\right)}$$

With

$$\begin{aligned} \cos\left(\frac{2\pi i}{n}\right) &> \cos\left(\frac{2\pi i}{n-1}\right) \quad i = 1, \dots, \frac{n}{2} - 1; \\ \cos\left(\frac{2\pi i}{n}\right) &> \cos\left(\frac{2\pi(i-1)}{n-1}\right), \quad i = \frac{n+1}{2}, \dots, n-1, \end{aligned}$$

we get $\frac{\partial \ln g(r, n)}{\partial r} > 0$. So $\ln g(r, n)$ is monotonically increasing with r . $\forall r, \forall n$, we have

$$\begin{aligned} \ln g(r, n) &> \lim_{r \rightarrow 0} \ln g(r, n) \\ &= \sum_{i=1}^{n-1} \ln\left(1 - \cos\left(\frac{2\pi i}{n}\right)\right) - \sum_{i=1}^{n-2} \ln\left(1 - \cos\left(\frac{2\pi i}{n-1}\right)\right) - \ln 2 \end{aligned}$$

Assume

$$h(n) = \sum_{i=1}^{n-1} \ln\left(1 - \cos\left(\frac{2\pi i}{n}\right)\right) - \sum_{i=1}^{n-2} \ln\left(1 - \cos\left(\frac{2\pi i}{n-1}\right)\right)$$

We have

$$\begin{aligned} h(n) - h(n-1) &= \sum_{i=1}^{n-1} \ln\left(2 \sin^2 \frac{\pi i}{n}\right) - 2 \sum_{i=1}^{n-2} \ln\left(2 \sin^2 \frac{\pi i}{n-1}\right) + \sum_{i=1}^{n-3} \ln\left(2 \sin^2 \frac{\pi i}{n-2}\right) \\ &= \sum_{i=1}^{n-1} \ln\left(\sin \frac{\pi i}{n}\right) - 2 \sum_{i=1}^{n-2} \ln \sin \frac{\pi i}{n-1} + \sum_{i=1}^{n-3} \ln \sin \frac{\pi i}{n-2} \\ &= \ln \frac{\prod_{k=1}^{n-1} \sin \frac{k\pi}{n} \cdot \prod_{k=1}^{n-3} \sin \frac{k\pi}{n-2}}{\prod_{k=1}^{n-2} \sin \frac{k\pi}{n-1} \cdot \prod_{k=1}^{n-2} \sin \frac{k\pi}{n-1}} \end{aligned}$$

According to $\prod_{k=1}^{n-1} \sin \frac{k\pi}{n} = \frac{n}{2^{n-1}}$, we have:

$$h(n) - h(n-1) = \ln \frac{n^2 - 2n}{n^2 - 2n + 1} < 0$$

So $h(n)$ is monotonically decreasing with n . Then we have

$$\begin{aligned}
& \ln g(r, n) > \lim_{r \rightarrow 0} g(r, n) = h(n) - \ln 2 \\
& > \lim_{n \rightarrow \infty} \left(\sum_{i=1}^{n-1} \ln \left(1 - \cos \left(\frac{2\pi i}{n} \right) \right) - \sum_{i=1}^{n-2} \ln \left(1 - \cos \left(\frac{2\pi i}{n-1} \right) \right) \right) - \ln 2 \\
& = \frac{1}{2\pi} \int_0^{2\pi} \ln(1 - \cos x) dx - \ln 2 \\
& = -2 \ln 2
\end{aligned}$$

Therefore,

$$\ln f(r, n) - \ln(2+r) + \ln(1+r) > -2 \ln 2$$

and

$$f(r+n) > \exp(-2 \ln 2 + \ln \frac{2+r}{1+r}) > \frac{1}{4}$$

■

F COMMUNICATION OVERHEAD ANALYSIS

Let n, l, W denote the number of instances, splitting candidates, and noise vectors per splitting candidate, respectively. For each node split, the communication overhead of different methods is:

- **Other DP methods:** $(2n + l)$ messages.
- **MaskedXGBoost:** $(lW + 2l)n + l$ messages.
- **HE method:** $(2n + 2l) \times ct$ messages, where ct is the ciphertext size.

For $n = 1000, l = 32, W = 3$, and HE method (Paillier) using 4096 bits encryption, the overhead is 16KB, 1.2MB, 2.1MB respectively. Our method has a lower overhead than the HE method but is still higher than other DP methods.

G ADVERSARY ANALYSIS

We further provide analysis on adversaries deviating from the protocol, building upon the honest-but-curious results in Section 6.5 and Section 6.6.

The protocol involves three components: the PP constructs the noise vectors (Algorithm 1), the AP adds noise based on these vectors (Algorithm 2), and the AP calculates the splitting score to determine the optimal split (Algorithm 3).

- The σ_2 component of the noise vector constructed by the PP protects its own privacy (as demonstrated in Theorem 3). However, as indicated by Theorem 1, increasing σ_2 leads to a degradation of the model's utility. If the PP deviates from the protocol by reducing σ_2 , it will expose more privacy; conversely, increasing σ_2 would compromise the goal of improving utility.
- For the AP, noise is added according to the parameter C . If the AP deviates from the protocol by reducing C , the privacy protection of its gradients will be weakened (as per Theorem 2). Increasing C would introduce additional disturbing noise into the vector, thereby reducing the utility, as to Theorem 1.
- If the AP deviates from the protocol in Algorithm 3, it will fail to identify the optimal split, leading to a decrease in the model's utility.

In our vertical federated learning setting with two parties, if either party violates the protocol, it will only affect its own privacy and will not improve its ability to launch a privacy attack against the other party. Additionally, it will reduce the model's utility.

H ADDITIONAL EXPERIMENTAL RESULTS

Figure 12 illustrates the utility evaluation results on Credit 2 and Nomao datasets. Figure 11 is the convergence evaluation results on Adult, Higgs, Bank, Credit 2, and Nomao datasets. Figure 13 is the empirical privacy evaluation results for AP on Credit 2 and Nomao datasets. Figure 14 is the empirical privacy evaluation results for PP on Credit 2 and Nomao datasets. Figure 15 shows the ablation study results on Adult, Higgs, Bank, Credit 2, and Nomao datasets.

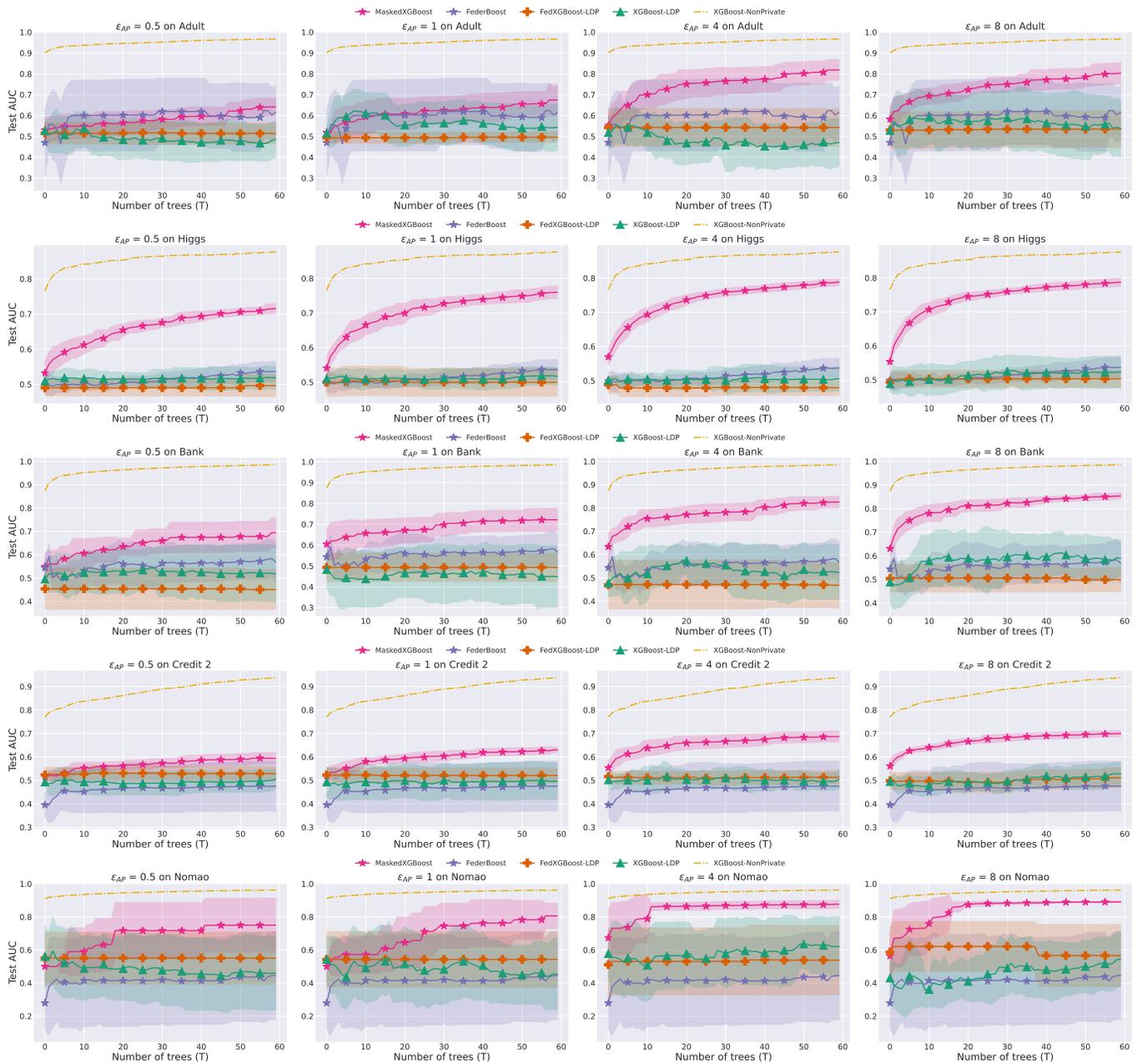


Figure 11: Training trees process of the four XGBoost algorithms with different privacy budgets on Adult, Higgs, Bank, Credit 2, and Nomao datasets.

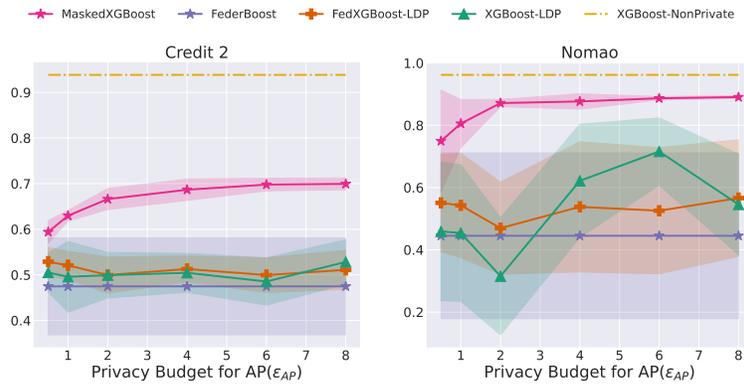


Figure 12: Utility of different methods for different privacy budgets on Credit 2 and Nomao.

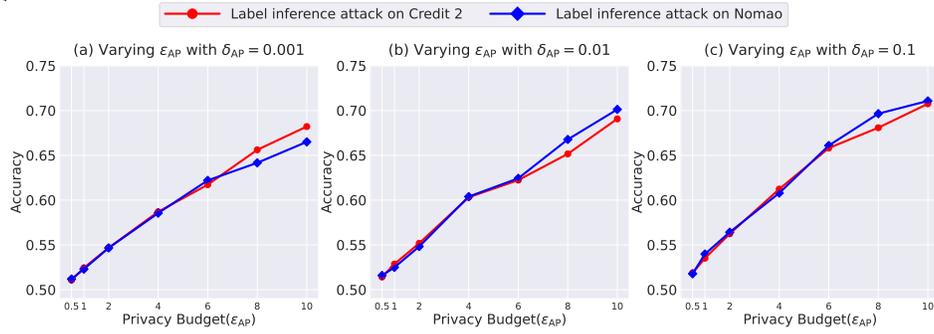


Figure 13: Empirical privacy evaluation for AP with varying privacy budget ϵ_{AP} and δ_{AP} . This is evaluated on Credit 2 and Nomao.

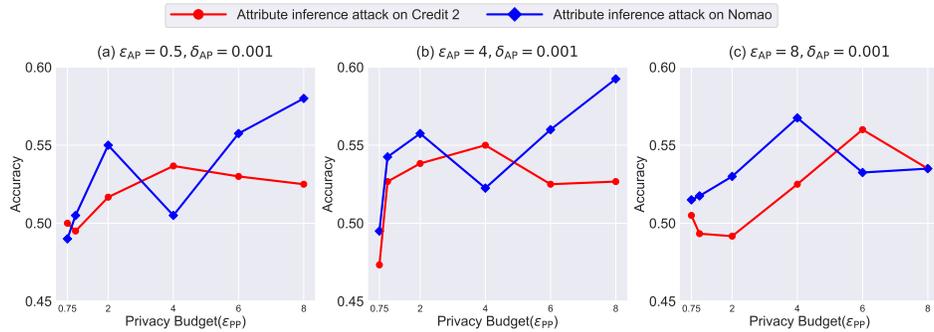


Figure 14: Empirical privacy evaluation for PP with varying privacy budget ϵ_{PP} and ϵ_{AP} . This is evaluated on Credit 2 and Nomao.

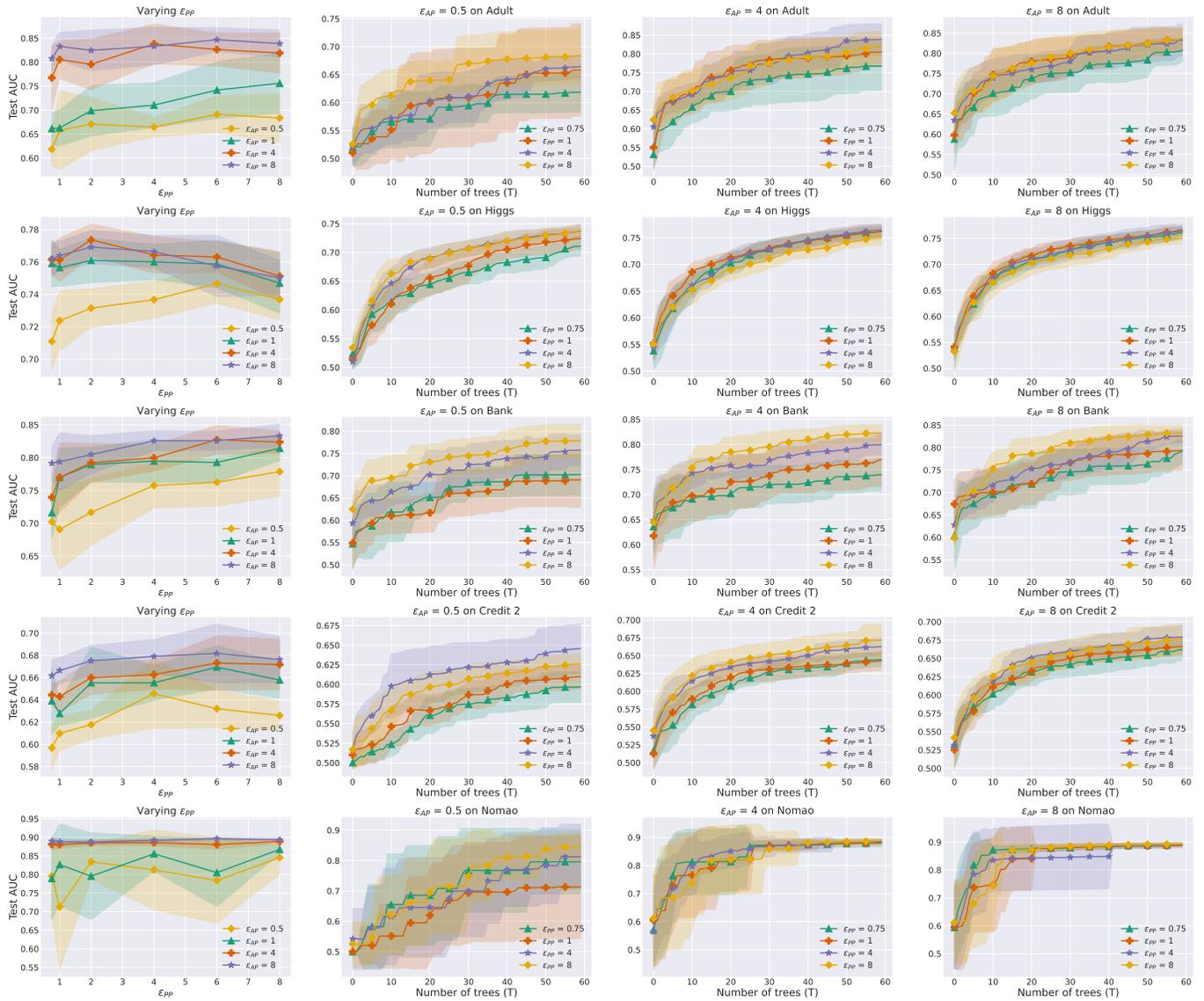


Figure 15: Ablation study on Adult, Higgs, Bank, Credit 2, and Bank datasets: the effect of ϵ_{PP} on final test AUC and training process.