FEVERLESS: FAST AND SECURE VERTICAL FEDERATED LEARNING BASED ON XGBOOST FOR DECENTRALIZED LABELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vertical Federated Learning (VFL) enables multiple clients to collaboratively train a global model over vertically partitioned data without revealing private local information. Tree-based models, like XGBoost and LightGBM, have been widely used in VFL to enhance the interpretation and efficiency of training. However, there is a fundamental lack of research on how to conduct VFL securely over distributed labels. This work is the first to fill this gap by designing a novel protocol, called FEVERLESS, based on XGBoost. FEVERLESS leverages secure aggregation via information masking technique and global differential privacy provided by a fairly and randomly selected noise leader to prevent private information from being leaked in the training process. Furthermore, it provides label and data privacy against honest-but-curious adversary even in the case of collusion of $n-2$ out of n clients. We present a comprehensive security and efficiency analysis for our design, and the empirical experiment results demonstrate that FEVERLESS is fast and secure. In particular, it outperforms the solution based on additive homomorphic encryption in runtime cost and provides better accuracy than the local differential privacy approach[1].

1 INTRODUCTION

Traditional centralized deep learning models, demanding to collect a considerable amount of clients' data to maintain high accuracy, to some degree, may increase the risk of data breaches. Data may not be easily shared among different entities due to privacy regulations and policies. To tackle this "Data Island" problem (Yang et al. (2019a)), Google proposed Federated Learning (FL) (McMahan et al. (2017)) to allow multiple clients to train a global model without sharing private data. The basic paradigm of FL is that all clients train local models with their own data, and then the information of local models, e.g., gradients, may be exchanged to produce a global model.

Based on different types of data partition (Yang et al. (2019a)), FL can be mainly categorized into Horizontal Federated Learning (HFL) and Vertical Federated Learning (VFL). The former focuses on training with horizontally partitioned data where clients share the same feature space but differing in data index set. Several research works (Shokri & Shmatikov (2015); Orekondy et al. (2019); Geiping et al. (2020); Li & Han (2019)) have found that training data of HFL is still at high risk of leakage in spite of private data is kept locally. Other studies (Phong et al. (2018); Truex et al. (2019); Xu et al. (2019); Zhang et al. (2020); Zhu et al. (2020)) were dedicated to enhancing the security of HFL. On the contrary, VFL is mainly applied in the scenario of training with vertically partitioned data (Wu et al. (2020); Cheng et al. (2021)) where clients share the same data index set but differing in feature space. In this paper, our principal focus is to achieve privacy-preserving training on VFL.

To best of our knowledge, many existing studies (Hardy et al. (2017); Nock et al. (2018); Liu et al. (2020); Yang et al. (2019b); Cheng et al. (2021); Chen & Guestrin (2016); Wu et al. (2020)) have proposed innovative approaches to prevent private information breaches in the context of VFL. Specifically, (Hardy et al. (2017)) introduced encryption-based privacy-preserving logistic regression to safeguard the information of data indexes. (Nock et al. (2018)) gave a comprehensive discussion on

[1]Code is available at: https://github.com/feverless111/vfl

the impact of ID resolution. Later, (Liu et al. (2020)) proposed an asymmetrically VFL scheme for logistic regression tackling privacy concerns on ID alignment. (Yang et al. (2019b)) introduced a scheme without using a coordinator for a limited number of clients.

Unlike the training models used in the aforementioned works, XGBoost (Chen & Guestrin (2016)), which is one of the most popular models applied in VFL, can provide better interpretation, easier parameter tuning, and faster execution than deep learning in tabular data training (Goodfellow et al. (2016); LeCun et al. (2015)). These practical features and advantages draw academia and industry's attention to the research on XGBoost, especially in the privacy-preserving context. (Wu et al. (2020)) introduced an approach for tree-based model training through a hybrid method composing homomorphic encryption and secure Multi-Party Computation (MPC) (Goldreich (1998); Bonawitz et al. (2017)). After that, (Cheng et al. (2021)) proposed a similar system to train XGBoost (Chen & Guestrin (2016)) securely over vertically partitioned data by using Additively Homomorphic Encryption (AHE). By applying Differential Privacy (DP) (Dwork (2008)), (Tian et al. (2020)) designed a VFL system to train GBDT without the need of encryption/decryption.

However, most of the above solutions based on AHE and MPC are shortfall in terms of efficiency on training XGBoost. Beyond that, all the existing schemes basically assume that training labels are managed and processed by a *sole* client. In practice, *a VFL scheme supporting distributed labels is of necessity*. For instance, multiply hospitals, clinics and health centers currently may be set to COVID-19 test spots and aim to train a model, e.g., XGBoost, to predict with good interpretation if citizens (settling in various locations) are infected based on their health records and symptoms. In this context, the labels (i.e., the test results) are very likely distributed among different health authorities. For example, a health clinic may collect and manage each of its registered patient's label locally. Another example may be a common scenario in the financial sector where multiple bank branches prefer to build a global model to predict if their customers may pay loans on time. The customers are registered to distinct branches, and branches could have various features, e.g., account balance, withdraw data. It is clear that labels must be distributed rather than centralized. In addition to efficiency and functionality aspects, one may also consider capturing stronger security for VFL. Training an XGBoost usually should involve the computation of first and second-order derivatives of the loss function (gradients and hessians containing labels' information), and aggregation of them is required in every round. In the setting of labels held by different clients, if gradients and hessians are transmitted in the form of plaintexts and summations of them are known by an aggregator (whom could be one of the clients engages in training), inference and differential attacks (Appendix C) can be conducted by the aggregator, resulting in the leakage of labels.

To tackle these problems, we propose a fast and secure VFL protocol, FEVERLESS, to train XGBoost (Appendix B.1) on distributed labels without disclosing both feature and label information. In our design, the privacy protection is guaranteed by secure aggregation (based on a masking scheme) and Global Differential Privacy (GDP) (Appendix B.6). We leverage masking instead of heavy-cost multiparty computation scheme and we can guarantee a "perfect secrecy" level for the masked data. In GDP, we use Verifiable Random Function (VRF) (Appendix B.5) to select a noise leader per round (who cannot be predicted and pre-compromised) to inject a global rather than accumulated noises (from all clients) to the computation results, which significantly maintains model accuracy.

Our contributions can be summarized as follows.
(1) We define VFL in a more practical scenario where training labels are distributed over multiple clients. Beyond that, we develop FEVERLESS to train XGBoost securely and efficiently with the elegant combination of secure aggregation technique (based on Diffie-Hellman (DH) key exchange (Appendix B.2) and Key Derivation Function (KDF) (Appendix B.4)) and GDP.
(2) We give a comprehensive security analysis to demonstrate that FEVERLESS is able to safeguard labels and features privacy in the semi-honest setting, but also maintain the robustness even for the case where $n - 2$ out of n clients commit collusion.
(3) We implement FEVERLESS and perform training time and accuracy evaluation on different real-world datasets. The empirical experiment results show that FEVERLESS can maintain efficiency and accuracy, which are comparable to the baseline - a "pure" XGBoost without using any encryption and differential privacy. Specifically, training the credit card and bank marketing datasets

just takes 1% and 6.5% more runtime than the baseline and meanwhile, the accuracy is only lower than that of the baseline by 0.9% and 3.21%, respectively[2].

2 PROBLEM FORMULATION

2.1 SYSTEM MODEL

Before proceeding, we give some assumptions on our model. We suppose that a private set intersection (Kolesnikov et al. (2017); Pinkas et al. (2014)) has been used to align data IDs before the training starts, so that each client shares the same data index space $\mathcal{I}$. But the names of features are not allowed to share among clients. As for the information of label distribution (indexes representing the relationship between labels and clients, e.g., a client A holds a label a), we will consider the following conditions: (1) this information is revealed to the public in advance; or (2) the information is not allowed to publish but the training can still be accomplished (with extra cost).

We also consider that the training is conducted on a dataset with m samples composing with feature space $\mathcal{X} = \{x_1, \cdots, x_m\}$, each containing d features, and label space $\mathcal{Y} = \{y_1, \cdots, y_m\}$. Besides, features $\{X_j^{(c)} \mid j \in \{1, \cdots, f\}\}$ and labels $\{y_i^{(c)} \mid i \in \{1, \cdots, m\}\}$ are held among n clients where each client has at least one feature and one label.$X_j^{(c)}$ and $y_i^{(c)}$ refer to j-th feature and i-th label owned by c-th client, respectively.

Considering a practical scenario wherein training labels are distributed among clients, we propose a new variant of VFL, named VFL over Distributed Labels (DL-VFL). The concrete definition is given as follows.

Definition 1 (DL-VFL)**.** *Given a training set with m data samples consisting of feature space $\mathcal{X}$, label space $\mathcal{Y}$, index space $\mathcal{I}$ and clients set $\mathcal{C}$, we have:*

$$\mathcal{X}^c \cap \mathcal{X}^{c'} = \emptyset, \mathcal{Y}^c \cap \mathcal{Y}^{c'} = \emptyset, \mathcal{I}^c = \mathcal{I}^{c'}, \forall c, c' \in \mathcal{C}, c \neq c'. \tag{1}$$

A client c participating DL-VFL shares the same sample ID space $\mathcal{I}$ with the corresponding labels, where a single label is only associated with a client. And different clients hold the subset of $\mathcal{X}$ sampled from feature space. To achieve privacy-preserving XGBoost training, we further define two roles as follows.

Definition 2 (Source client)**.** *A source client with split candidates would like to compute the corresponding L_{split} based on Eq.(4). But some labels are missing so that $\sum g_i$ and $\sum h_i$ are unable to derive.*

For the case that a source client does not hold all labels in the current split candidate, we propose a solution based on secure aggregation and global differential privacy to help the source client to compute L_{split} while safeguarding other clients' privacy. We consider the two conditions regarding if label distribution is publicly known. We find that if we keep label distribution hidden, then we will take extra communication overheads to perform training. The detailed explanation can be seen in Appendix F. Note each client may have a chance to act as a source client because all the labels are distributed in our setting, where the *source client* leads the L_{split} computation, and *client* provides missing label values to the source client.

To achieve GDP where there is only one client that injects noise per training round, we define noise leader who is selected fairly and randomly from all clients (except the source client) - preventing clients from being compromised beforehand.

Definition 3 (Noise leader)**.** *By using VRF, a noise leader is responsible for generating the maximum leader score and adding differentially private noise to the aggregated values.*

Note we summary the main notations used in this paper in Table 1 (see Appendix A).

[2]For banknote authentication dataset, FEVERLESS takes 13.96% more training time than the baseline, and the accuracy is 30.4% lower. This is because the model is trained by a small-scale dataset, so that the robustness is seriously affected by noise.

2.2 THREAT MODEL

We mainly consider potential threats incurred by participating clients and the outside adversaries. We assume that all clients are *honest-but-curious*, which means they strictly follow designed algorithms but try to infer private information of other clients from the received messages. Besides, we also consider up to $n-2$ clients' collusion to conduct attacks. Through authenticated channels, DH key exchange can be securely executed among clients. Other messages are transmitted by public channels, and outside attackers can eavesdrop on these channels and try to reveal information about clients during the whole DL-VFL process. Note this paper mainly focuses on solving privacy issues in training DL-VFL based on XGBoost. Thus, other attacks, like data poisoning and backdoor attacks deteriorating model performance, are orthogonal to our problem.

3 A PRACTICAL PRIVACY-PRESERVING PROTOCOL

3.1 FEVERLESS PROTOCOL DESCRIPTION

To prevent a source client from knowing gradients and hessians sent by other clients, one may directly use MPC (Damgård et al. (2012)) based on AHE (Paillier (1999); Wu et al. (2020)). This seems to be a trivial solution. But this method yields excessive computation costs. Getting rid of the complex system like MPC, we leverage secure aggregation protocol via masking scheme based on DH key exchange(Bonawitz et al. (2017); Ács & Castelluccia (2011); Tian et al. (2020)). By further using KDF and Hash Function (see Appendix B.3&B.4), our masking (for gradients and hessians) can be derived without exchanging keys per training round. Our approach significantly reduces the communication cost but still maintains the robustness up to $n-2$ colluded clients. Meanwhile, the secure aggregation can provide "perfect secrecy" for broadcast messages. After receiving the broadcast messages, the masking will be canceled out at the source client side. In this case, only using the masking scheme is unable to defend against differential attacks. We thus turn our attention to Local Differential Privacy (LDP) (Kairouz et al. (2014)), so that each client may add noise to per send-out message, barely consuming any extra computation cost. But the accumulated noise, from all clients, may seriously affect the model accuracy. In our design, we consider a GDP (Wei et al. (2020)) approach with noise leader selection to avoid this shortcoming. Specifically, we use VRF to randomly select one client (each round) to securely inject a global noise, so that this client will not be predicted and set beforehand - reducing its chance to be corrupted by outsiders and the source client. Therefore, a hybrid method is formed based on masking scheme and GDP, so that per client's sensitive information can be protected by the "masks" and the aggregated values are secured by the noise which is injected by the chosen client.

Below we briefly introduce the main steps of our design, and the detailed algorithms and more explanations can be found in Appendix D. Assume each client $c \in [1, n]$ generates respective secret key sk_c and computes gradients $g_i^{(c)}$ and hessians $h_i^{(c)}$ locally, where $\{i \mid y_i \in \mathcal{Y}^c\}$. FEVERLESS works as follows.

1. *Broadcast missing indices.* The source client broadcasts the missing data indices mIDs$= \{i \mid y_i \notin \mathcal{Y}^c\}$.

2. *Key exchange computation.* Each client c computes public key $pk_c = g^{sk_c}$ using secret keys sk_c, sends pk_c to other clients and computes the corresponding shared keys[3] $\{S_{c,c'} = pk_{c'}^{sk_c} = g^{sk_c sk_{c'}} \mid c, c' \in \mathcal{C}, c \neq c'\}$ based on secret key sk_c received public keys $\{pk_{c'} \mid c' \in \mathcal{C}\}$.

3. *Data masking.* Each client c runs the masking generation algorithm to compute the maskings for protecting gradients and hessians. Specifically, based on KDF, clients' indices and the number of queries, the masking generation algorithm are conducted by $\texttt{mask}_g^{(c)} \leftarrow \sum_{c \neq c'} \frac{|c-c'|}{c-c'} \cdot \Big(\mathsf{H}(S_{c,c'} \| 0 \| \texttt{query})\Big)$, $\texttt{mask}_h^{(c)} \leftarrow \sum_{c \neq c'} \frac{|c-c'|}{c-c'} \cdot \Big(\mathsf{H}(S_{c,c'} \| 1 \| \texttt{query})\Big)$[4]. Then masked gra-

[3]Shared keys are only generated once, and the KDF is used to generate the remaining maskings.

[4]For purpose of simplicity, we omit modular computations. The complete calculation processes are elaborated on Algorithm 3.

dients $G^{(c)}$ and hessians $H^{(c)}$ are generated by $G^{(c)} = \sum_{i \in mIDs} g_i^{(c)} + \texttt{mask}_g^{(c)}$, $H^{(c)} = \sum_{i \in mIDs} h_i^{(c)} + \texttt{mask}_h^{(c)}$.

4. *Noise leader selection.* Each client generates the selection score $selec_c$ using the VRF, $\mathsf{H}(\texttt{SIGN}_{sk_c}(\texttt{count}))$, and broadcasts it, where `count` is the number of times clients conducting VRF, and `SIGN` is the signature scheme (see Appendix B.5 for more details). The client with maximum score will become the noise leader. For ease of understanding, we assume that client n has the largest selection score $select_n^{max}$ and is selected as the noise leader, in Figure 1.

5. *Global noise injection.* The leader n adds the noise $N(0, \Delta_g^2\sigma^2)$ and $N(0, \Delta_h^2\sigma^2)$ to $G^{(n)}$ and $H^{(n)}$ respectively.

6. *Aggregation and computation.* All clients send the masked values to the source client. The source client computes $\sum_{c=1}^{n} G^{(c)} + N(0, \Delta_g^2\sigma^2)$, $\sum_{c=1}^{n} H^{(c)} + N(0, \Delta_h^2\sigma^2)$ and L_{split}.

7. *Final update.* The source client with maximum L_{split} updates model following XGBoost (Chen & Guestrin (2016)) and broadcasts updated model and data indices in child nodes as step 8.

Figure 1 gives a high level overview of FEVERLESS. Note this process can be conducted iteratively. For simplicity, the core calculation processes are shown here, and more details are in Appendix D.

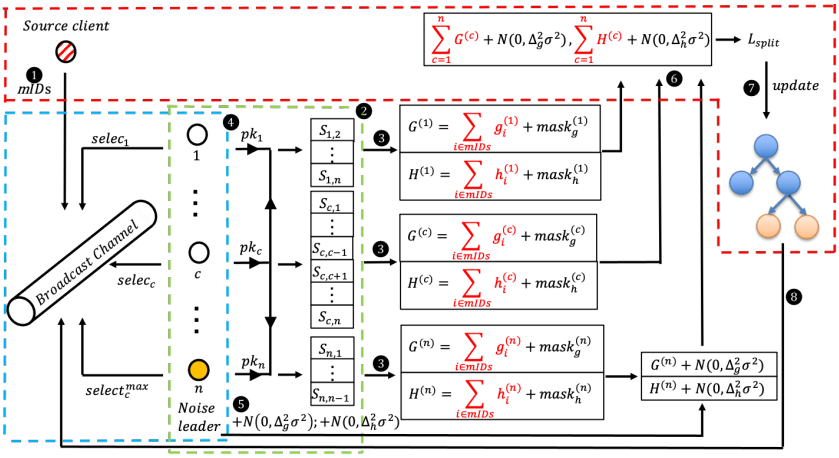

Figure 1: Overview of FEVERLESS. - - - - : DH key exchange and maskings generation. - - - - : Noise leader selection. - - - - : Source client broadcasts missing IDs, aggregates gradients and hessians securely, updates model and broadcasts nodes IDs. ❶Broadcast missing indices. ❷Key exchange computation. ❸Data masking. ❹Noise leader selection. ❺Global noise injection. ❻Aggregation and computation. ❼❽ Final update and broadcast updated model. Note sensitive data are in red. The maskings in ❸ protect data from source client, and the noise in aggregated gradients and hessians prevents source client from conducting differential attack.

3.2 Theoretical Analysis

Computation cost: We use B and d to denote the number of buckets and the maximum depth respectively, and $f^{(c)}$ here represents the number of features held by a client c. For each client c, the computation costs can be divided into 4 parts: (1) Performing at most $f^{(c)} \cdot B \cdot NT \cdot (2^d - 1)$ times computation of L_{split} and w, taking $O(f^{(c)} \cdot B \cdot NT \cdot 2^d)$ time; (2) Creating $n-1$ shared keys and 1 public key, which is $O(n)$; (3) Conducting $O(f^{(c)} \cdot B \cdot NT \cdot 2^d)$ time to compute VRF outputs and select noise leader; (4) Generating $2f^{(c)} \cdot B \cdot NT \cdot (2^d - 1)$ maskings, which takes $O(f^{(c)} \cdot B \cdot NT \cdot 2^d \cdot n)$ time. Overall, each client's computation complexity is $O(f^{(c)} \cdot B \cdot NT \cdot 2^d + n)$.
Communication cost: Each client's communication costs can be calculated as (1) Broadcasting at most $f^{(c)} \cdot B \cdot NT \cdot (2^d - 1)$ times of missing indexes mID; (2) Broadcasting 1 public key and receiving $n-1$ public keys from other clients; (3) Broadcasting 1 leader selection score at most $f^{(c)} \cdot B \cdot NT \cdot (2^d - 1)$ times, (4) Sending source client 2 masked gradients and hessians of size

$2\lceil log_2 N \rceil$. Therefore the overall communication cost is $f^{(c)} \cdot B \cdot NT \cdot (2^d - 1) \cdot (\|mID\| \cdot \alpha_I + \alpha_L + n \cdot \alpha_K 2\lceil log_2 N \rceil)$, where α_I, α_L and α_K refer to the number of bits of index, leader selection score and public keys, respectively. Thus, we have the communication complexity $O(f^{(c)} \cdot B \cdot NT \cdot 2^d)$.

3.3 Security Analysis

We prove that FEVERLESS provides label and data privacy against an adversary controlling at most $n - 2$ clients in the *semi-honest* setting (Smart (2016)). Here, we provide a brief summary of our analysis, and the formal proofs, in the random oracle model, are given in Appendix E.

Label Privacy: Label privacy implies that the owner of a label among honest parties should not be leaked to the adversary. We achieve this property by using a secure aggregation mechanism where the masks are created via DH key exchange (Diffie & Hellman (1976)) and KDF. In brief, we show that because of the Decisional DH problem (see Definition 4), the adversary cannot distinguish the individual values from randomly chosen ones. That is why the adversary cannot learn the owner of the label.

Data Privacy: Our protocol provides data privacy, meaning that an adversary cannot extract the data of any honest party. Individual data values are not separable from random values because of the secure masking. If the source client is not part of the adversary, no data information is leaked. However, we require an additional countermeasure for the case where the source client is part of the adversary because it can collect the summation of the data values. We use differential privacy Dwork et al. (2006b;a) to achieve data privacy. Because of the noise added by differential privacy, the adversary cannot learn the individual data of an honest client. Moreover, we select the noise client by the VRF which ensures that the noise leader cannot be predicted or compromised in advance.

4 Experiment

We perform evaluations on accuracy and runtime performance, and compare our design with two straightforward secure approaches: one is based on LDP (for accuracy), and the other is built on AHE with GDP (for runtime). These approaches are most-commonly-used components for secure solutions to FL, and they could be used as building blocks to form complex mechanisms, e.g., MPC. We note that the protocol should intuitively outperform those MPC-based solutions, and one may leverage our source code to make further comparison if interested. In the experiments, the baseline, which is the pure XGBoots algorithm, follows the training process of Figure 1 without using any privacy-preserving mechanisms (steps ❷ - ❺). And LDP does not conduct DH key exchange but each client injects noise into the aggregation of gradients and hessians, while AHE follows Figure 1 except executing DH key exchange. Besides, in AHE, each client sends (additive) encrypted messages to source client after step ❺.

4.1 Experiment Setup

All the experiments are implemented in Python, and conducted on a cluster of machines with Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, with 15GB RAM in a local area network. To investigate the impact of noise on accuracy, we set ϵ to be: 10, 5, 2 and 1, and δ is set to 10^{-5}. Intuitively, the smaller ϵ we set, our system will become more secure; but, at the same time, the larger noise will be added. We note that the above statement can be seen in the experimental results. As for the setting of cryptographic tools, we generally set the key size of DH and Paillier as 1024 bits (so as to save some time in running the experiments). This size is already able to reach a symmetric security level with 80 bits key length. Note one may indeed increase the key size to obtain a stronger security level[5], but this will bring a longer experiment time as a side effect. We use 1024-bit MODP Group with 160-bit Prime Order Subgroup from *RFC 5114* [6] for DH Key exchange. Further, `SHAKE-256` (Dworkin (2015)), a member of `SHA3` (Dworkin (2015)) family, producing an arbitrary bit length output, will be used as a hash function in leader selection and secure aggregation.

In terms of accuracy and runtime, we evaluate different solutions by varying the number of clients, the number of trees, the maximum depth of trees, all from 2 to 10. Other parameters regarding to

[5]Note that a stronger security level will not affect the training accuracy.

[6]https://tools.ietf.org/html/rfc5114

training follow the suggestions given in (Chen & Guestrin (2016)) and the *library* [7] of XGBoost. To deliver fair results, we conduct each test for 20 independent trials and then calculate the average.

Datasets. We run the experiments on three datasets - Credit Card (Yeh & Lien (2009)), Bank Marketing (Moro et al. (2014)) and Banknote Authentication[8] - for classification tasks. To fairly investigate the model performance in DL-VFL, we make the labels as sparse as possible, and they are uniformally distributed on clients. We present the details of datasets in Appendix G.

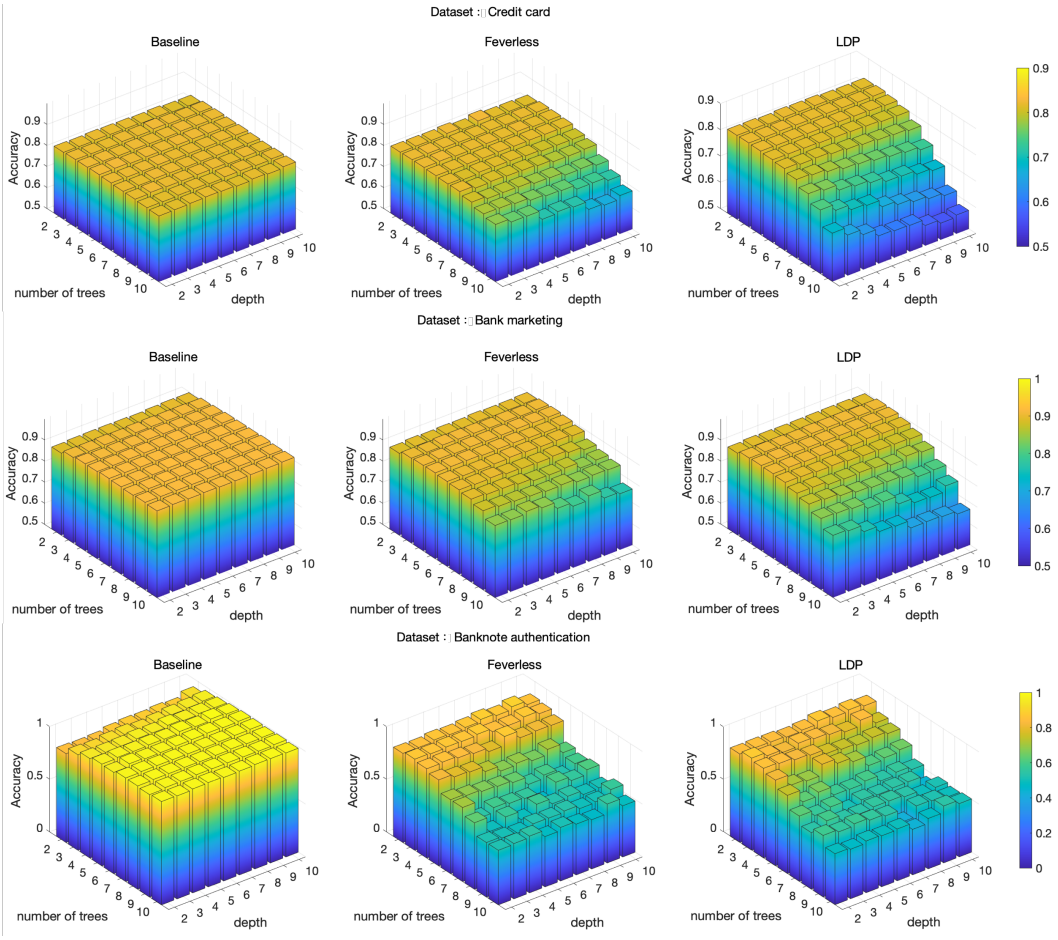

Figure 2: Comparison among the baseline, FEVERLESS and LDP under $\epsilon = 2$. *Top row:* Credit card dataset, accuracy range: [0.5, 0.9]. *Middle row:* Bank marketing, accuracy range: [0.5, 1]. *Bottom row:* Banknote authentication, accuracy range: [0, 1].

4.2 Evaluation on Accuracy

In Figure 2, we present a clear picture on how the accuracy varies from the number of trees and the maximum depth in $(2, 10^{-5})-$DP. We merge the client number in one tree structure, which means in one bar, and the value is the mean of accuracy when conducting on different numbers. The accuracy of the baseline in credit card (about 0.82) and bank marketing (nearly 0.9) remains unchanged as the number of trees and maximum depth increases, while the accuracy in banknote authentication rises from 0.9 to approximately 1.0. In order to highlight the differences and ensure that all results can be displayed clearly, we set the ranges of accuracy as $[0.5, 0.9]$, $[0.5, 1]$ and $[0, 1]$ for the three datasets, respectively. We note that for those experiments related to the accuracy performance based on the increase of client number, please refer to Appendix H.

[7] https://xgboost.readthedocs.io/

[8] https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Comparing with the baseline, shown in the top and middle rows of the Figure 2, FEVERLESS and LDP suffer from continuously shrinking accuracy as tree structure becomes complex. This is so because the injected noises are accumulated into the model via the increase of query number. And the accuracy is easily affected by the depth. In the worst case where the numbers of trees and maximum depth are both equal to 10, FEVERLESS decreases 10.37% (resp. 14.98%), and LDP drops 24.78% (resp.24.59%) in credit card (resp. bank marketing). But on average, FEVERLESS' accuracy only shrinks by around 0.9% (resp. 3.21%), while LDP suffers from estimated 3x (resp. 2x) accuracy loss. The difference in the degree of deterioration mainly comes from the factor that how much noise is added for each query. We note the deterioration of FEVERLESS is independent with the number of clients in the training. This means that our design can maintain great accuracy even for the case where there exists a considerable amount of client. The more clients we have, the better accuracy we can see as compared to the LDP approach.

Despite the fact that less noise is added in FEVERLESS, we do not predict that the accuracy of FEVERLESS falls to the same level (around 50%, like randomly guess in binary classification) as LDP in the bottom row of Figure 2. An explanation for this is that the model is trained by an extremely small-size dataset, resulting in lack of robustness and sensitivity to noise. If we set a larger ϵ, the advantage of our scheme will be more clear. To keep the consistency of predefined parameters, separate experiments are not conducted on banknote authentication dataset with different ϵ.

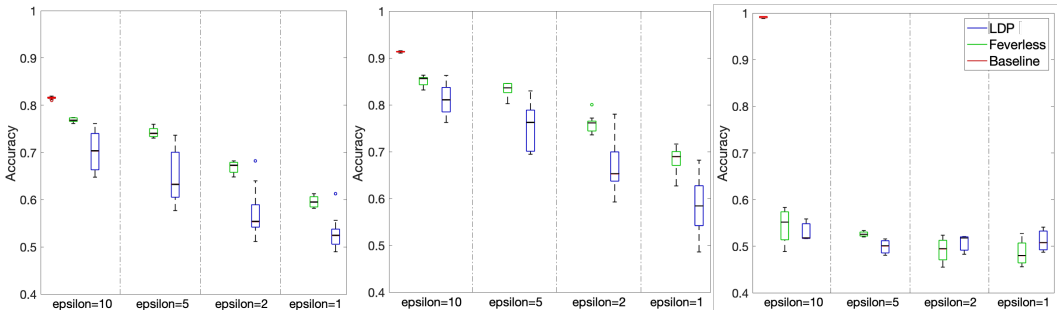

Figure 3: Comparison of accuracy by varying ϵ in depth=10, the number of trees=10. *Left:* Credit card. *Middle:* Bank marketing. *Right:* Banknote authentication. Accuracy ranges from 0.4 to 1.

To distinguish the performance between FEVERLESS and LDP more clearly, Figure 3 shows the comparison over different ϵ, when #depth and #tree are set to 10. The performance of the model is decayed as the decrease of ϵ. In the left (resp. middle) of the Figure 3, the averaged accuracy of FEVERLESS falls from 0.7686 to 0.5967 (resp. from 0.8517 to 0.6831), while that of LDP also decreases to 0.5299 (resp. 0.5853). We notice that the highest values of LDP stay at the same level as those of FEVERLESS. This is because, in the case of 2-client training, only one client needs to add the noise in LDP (which is identical to our GDP solution). At last, the worse case can be seen in the right of the Figure 3 due to the weak robustness of the model obtained from the banknote authentication. The results are much far away from the baseline there. But even in this case, FEVERLESS still holds a tiny advantage over LDP.

4.3 Evaluation on Training Time

To highlight the runtime complexity, we average the results varying by client number into one tree structure as well. We further set the ranges of time as [0s, 9,500s], [0s, 3,500s] and [0s, 110s] for the datasets to deliver visible results. Note since the banknote dataset contains the least samples, it does deliver the best training efficiency here. Figure 4 presents the comparison on the training time by varying maximum depth and the number of trees among the datasets.

The training time increases exponentially and linearly with depth and the number of tree, which is consistent with our analysis given in Section 3.2. In Figure 4, compared with the baseline, the runtime of FEVERLESS at most increases 110.3s (resp. 50s, 4.3s), while AHE requires around 70x spike (resp. 48x, 21x) in credit card (resp. bank marketing, banknote authentication), where #depth and #trees are equal to 10. For the average case, FEVERLESS consumes Approx. $1\%(resp.6.5\%, 13.96\%)$ more training time than the baseline, while AHE requires the

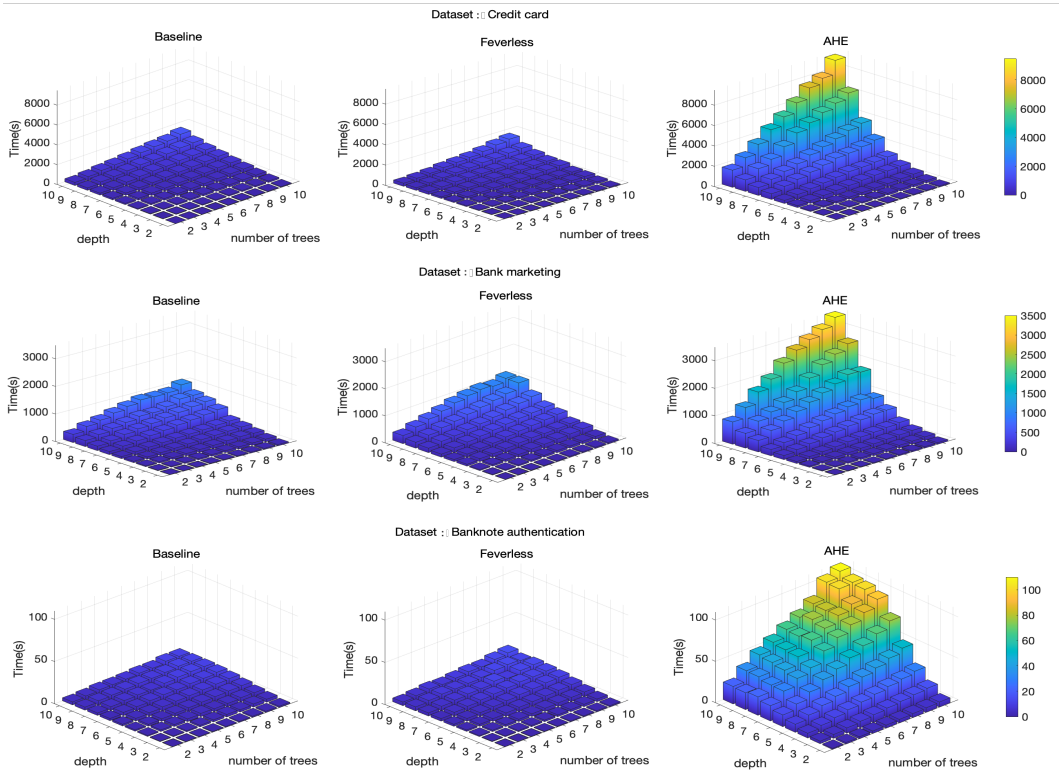

Figure 4: Comparison of time. *Top row:* Credit card dataset, range: [0s, 9,500s]. *Middle row:* Bank marketing, range: [0s, 3,500s]. *Bottom row:* Banknote authentication, range: [0s, 110s].

$351\%(resp.155.1\%, 674\%)$ extra, w.r.t. the three datasets. Its poor performances are due to the laborious calculations in encryption, in which each client has to conduct an encryption per query. By contrast, the masksings in FEVERLESS avoid these excessive costs. We further investigate the runtime performance on the increase of #client, and the results are in Appendix H.

5 Conclusion and Future Work

We consider a practical scenario where labels are distributedly maintained by different clients for VFL. By leveraging secure aggregation and GDP, we present a novel system, FEVERLESS, to train XGBoost securely. FEVERLESS can achieve perfect secrecy for label and data, and adversaries cannot learn any information about the data if the source client is not corrupted. With DP against differential attack, the source client knows nothing more than summation. Our design is also robust for the collusion of $n - 2$ out of n clients. The experiment results show that FEVERLESS is fast and accurate, only taking 1% extra training time, and sacrificing 0.9% accuracy, as compared to the baseline - a pure XGBoost.

Discussions. In the semi-honest setting, if the source client sends the missing indexes consistently, adversaries may figure out which labels are distributed (on the source clients) by statistical analysis. We show that this issue can be tackled, and please refer to the details in Appendix F. Besides, instead of adopting VRF in FEVERLESS, other secure mechanisms may be applied, such as Proof of Elapsed Time (PoET) (Chen et al. (2017); Corso (2019)), to maintain consensus and system security (also see Appendix F). Although our system achieves great performance in terms of security and efficiency, its accuracy still does not work well in small-scale datasets, like banknote authentication. This remains an open problem for our future work. And we will also consider secure solutions against malicious adversaries.

REFERENCES

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

G. Ács and C. Castelluccia. I have a dream! (differentially private smart metering). In *Information Hiding*, 2011.

Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet transform. *IEEE Transactions on image processing*, 1(2):205–220, 1992.

Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C-W Phan. Sha-3 proposal blake. *Submission to NIST*, 92, 2008.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pp. 1175–1191, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3133982. URL https://doi.org/10.1145/3133956.3133982.

Dan Boneh. The decision diffie-hellman problem. In *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pp. 48–63. Springer, 1998.

Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. On security analysis of proof-of-elapsed-time (poet). In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pp. 282–297. Springer, 2017.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Kewei Cheng, Tao Fan, Yilun Jin, Yang Liu, Tianjian Chen, Dimitrios Papadopoulos, and Qiang Yang. Secureboost: A lossless federated learning framework, 2021.

Amie Corso. *Performance analysis of proof-of-elapsed-time (poet) consensus in the sawtooth blockchain framework*. PhD thesis, University of Oregon, 2019.

Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*, pp. 643–662. Springer, 2012.

Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li (eds.), *Theory and Applications of Models of Computation*, pp. 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-79228-4.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 486–503. Springer, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006b.

Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. 2015.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients - how easy is it to break privacy in federated learning? In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 16937–16947. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c4ede56bbd98819ae6112b20ac6bf145-Paper.pdf.

Oded Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *Advances in neural information processing systems*, 27:2879–2887, 2014.

Burt Kaliski. *Pseudorandom Function*, pp. 485–485. Springer US, Boston, MA, 2005. ISBN 978-0-387-23483-0. doi: 10.1007/0-387-23483-7_329. URL https://doi.org/10.1007/0-387-23483-7_329.

Aqeel Sahi Khader and David Lai. Preventing man-in-the-middle attack in diffie-hellman key exchange protocol. In *2015 22nd international conference on telecommunications (ICT)*, pp. 204–208. IEEE, 2015.

Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1257–1272, 2017.

Hugo Krawczyk and Pasi Eronen. Hmac-based extract-and-expand key derivation function (hkdf). Technical report, RFC 5869, May, 2010.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Hongyu Li and Tianqi Han. An end-to-end encrypted neural network for gradient updates transmission in federated learning. *arXiv preprint arXiv:1908.08340*, 2019.

Yang Liu, Xiong Zhang, and Libin Wang. Asymmetrically vertical federated learning. *arXiv preprint arXiv:2004.07427*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research*, 17(1):1235–1241, 2016.

Silvio Micali. ALGORAND: the efficient and democratic ledger. *CoRR*, abs/1607.01341, 2016. URL http://arxiv.org/abs/1607.01341.

Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *40th annual symposium on foundations of computer science (cat. No. 99CB37039)*, pp. 120–130. IEEE, 1999.

Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

Richard Nock, Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Entity resolution and federated learning get a federated resolution. *arXiv preprint arXiv:1803.04035*, 2018.

Tribhuvanesh Orekondy, Seong Joon Oh, Yang Zhang, Bernt Schiele, and Mario Fritz. Gradient-leaks: Understanding and controlling deanonymization in federated learning. In *NeurIPS Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *TAMC*. Springer, 1999.

Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, 13(5):1333–1345, 2018. doi: 10.1109/TIFS.2017.2787987.

Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster private set intersection based on {OT} extension. In *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 797–812, 2014.

NIST Sha. standard: Permutation-based hash and extendable-output functions. federal information processing standards publication 202, 2015, 2015.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pp. 1310–1321, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813687. URL https://doi.org/10.1145/2810103.2813687.

Nigel P. Smart. *Cryptography Made Simple*. Information Security and Cryptography. Springer, 2016.

Zhihua Tian, Rui Zhang, Xiaoyang Hou, Jian Liu, and Kui Ren. Federboost: Private federated learning for gbdt, 2020.

Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, pp. 1–11, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368339. doi: 10.1145/3338501.3357370. URL https://doi.org/10.1145/3338501.3357370.

Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. Privacy preserving vertical federated learning for tree-based models. *Proc. VLDB Endow.*, 13(12):2090–2103, July 2020. ISSN 2150-8097. doi: 10.14778/3407790.3407811. URL https://doi.org/10.14778/3407790.3407811.

Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, pp. 13–23, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368339. doi: 10.1145/3338501.3357371. URL https://doi.org/10.1145/3338501.3357371.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), January 2019a. ISSN 2157-6904. doi: 10.1145/3298981. URL https://doi.org/10.1145/3298981.

Shengwen Yang, Bing Ren, Xuhui Zhou, and Liping Liu. Parallel distributed logistic regression for vertical federated learning without third-party coordinator. *arXiv preprint arXiv:1911.09824*, 2019b.

I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications*, 36(2): 2473–2480, 2009.

Jonathan Zdziarski. *Hacking and securing iOS applications: stealing data, hijacking software, and how to prevent it*. " O'Reilly Media, Inc.", 2012.

Chengliang Zhang, Suyi Li, Junzhe Xia, Wei Wang, Feng Yan, and Yang Liu. Batchcrypt: Efficient homomorphic encryption for cross-silo federated learning. In *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pp. 493–506. USENIX Association, July 2020. ISBN 978-1-939133-14-4. URL https://www.usenix.org/conference/atc20/presentation/zhang-chengliang.

Hangyu Zhu, Rui Wang, Yaochu Jin, Kaitai Liang, and Jianting Ning. Distributed additive encryption and quantization for privacy preserving federated deep learning. *arXiv preprint arXiv:2011.12623*, 2020.

A NOTATIONS

The frequently used notations are summarized in Table 1.

Table 1: Notations summary

Notation	Description
$\mathcal{X}$	feature space
$X_j^{(c)}$	j-th feature owned by c-th client
x_i	i-th data point with d features
$\mathcal{Y}$	label space
$y_i^{(c)}$	the label of i-th data point owned by c-th client
$\mathcal{I}$	data index space
$\mathcal{C}$	clients set
$g_i^{(c)}$	the gradient of i-th data point owned by c-th client
$h_i^{(c)}$	the hessian of i-th data point owned by c-th client
G	summation of gradients
H	summation of hessians
m	number of data entries
n	number of clients
f	number of features
d	the maximum depth of tree
ϵ, δ	parameters of differential privacy
Δ_g	sensitivity of gradients
Δ_h	sensitivity of hessians
L_{split}	impurity score
w	leaf value
pk_c	public key generated by c-th client
sk_c	secret key owned by c-th client
g	generator of multiplicative group
B_z^j	z-th bucket of j-th feature

B PRELIMINARIES

B.1 XGBOOST

XGBoost Chen & Guestrin (2016) is a popular tree-based model in tabular data training that can provide better interpretation, easier parameters tuning and faster execution speed than deep learning Goodfellow et al. (2016); LeCun et al. (2015). It also outperforms other well-known boosting tree systems in terms of accuracy and efficiency, like Spark MLLib Meng et al. (2016) and H2O Chen & Guestrin (2016), especially for large-scale datasets. Therefore, in this paper, we consider using XGBoost as a building block for classification tasks.

Assume that a training set with m data points composing with feature space $\mathcal{X} = \{x_1, \cdots, x_m\}$ and label space $\mathcal{Y} = \{y_1, \cdots, y_m\}$. Before training starts, every feature will be sorted based their values, and split candidates will be set for features. XGBoost builds trees based on the determination of defined splits candidates and some pruning conditions. Specifically, computing gradients and hessians first according to Eq.(2) and Eq.(3) for each data entry, where $y_i^{(t-1)}$ denotes the prediction of previous tree for i-th data point, and y_i is the label of i-th data point:

$$g_i = \frac{1}{1 + e^{-y_i^{(t-1)}}} - y_i = \hat{y}_i - y_i, \tag{2}$$

$$h_i = \frac{e^{-y_i^{(t-1)}}}{(1 + e^{-y_i^{(t-1)}})^2}. \tag{3}$$

For splitting nodes, the XGBoost algorithm determines the best split candidate from all others based on maximum L_{split} in Eq.(4), where λ and γ are regularization parameters:

$$L_{split} = \frac{1}{2}[\frac{\sum_{i \in I_L} g_i}{\sum_{i \in I_L} h_i + \lambda} + \frac{\sum_{i \in I_R} g_i}{\sum_{i \in I_R} h_i + \lambda} - \frac{\sum_{i \in I} g_i}{\sum_{i \in I} h_i + \lambda}] - \gamma. \tag{4}$$

The current node will be the leaf node if the following conditions are fulfilled: reaching the maximum depth of tree, the maximum value of impurity is less than preset threshold. The calculation of the leaf value follows Eq.(5):

$$w = -\frac{\sum_{i \in I} g_i}{\sum_{i \in I} h_i + \lambda}. \tag{5}$$

B.2 Diffie-Hellman key exchange

Based on Decision Diffie-Hellman (DDH) hard problem (Boneh (1998)) defined below, Diffie-Hellman key exchange (DH) (Diffie & Hellman (1976)) provides a method used for exchanging keys across public communication channels. Without losing generality and correctness, it consists of a tuple of algorithms (**Param.Gen, Key.Gen, Key.Exc**). The algorithm $(\mathbb{G}, g, q) \leftarrow$ **Param.Gen** (1^α) generates public parameters (a group $\mathbb{G}$ with prime order q generated by a generator g) based on secure parameter α. $(sk_i, pk_i) \leftarrow$ **Key.Gen**$(\mathbb{G},\ g,\ q)$ allows client i to generate secret key ($sk_i \xleftarrow{\$} \mathbb{Z}_q$) and compute public key ($pk_i \leftarrow g^{sk_i}$). Shared key is computed by $(pk_i^{sk_j}, pk_j^{sk_i}) \leftarrow$ **Key.Exc**(sk_i, pk_i, sk_j, pk_j). Inspired by (Bonawitz et al. (2017); Ács & Castelluccia (2011)), we utilize shared keys as maskings to protect information of labels against inference attack during transmitting in public channels. The correctness requires $pk_i^{sk_j} = pk_j^{sk_i}$. The security relies on the DDH problem (Boneh (1998)), which is defined as:

Definition 4 (Decision Diffie-Hellman)**.** *Let $\mathbb{G}$ be a group with prime order q and g be the fixed generator of the group. The Probabilistic Polynomial Time (*PPT*) adversary $\mathcal{A}$ is given and g^a and g^b where a and b are randomly chosen. The probability of $\mathcal{A}$ distinguishing (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) for a randomly chosen c is negligible:*

$$\Big|\Pr\Big[a, b \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(g, g^a, g^b, g^{ab}) = \mathsf{true}\Big] - \Pr\Big[a, b, c \xleftarrow{\$} \mathbb{Z}_q : \mathcal{A}(g, g^a, g^b, g^c) = \mathsf{true}\Big]\Big| < negl(\alpha).$$

B.3 Pseudo-random generator and hash function

Pseudo-Random Generator (PRG) (Håstad et al. (1999)) is an algorithm which is able to generate random numbers. The "pseudo-random" here means that the generated number is not truly random but has the similar properties with random number. Generally, the pseudo-random numbers are determined by given initial values a.k.a seeds. In cryptographic applications, a secure PRG requires attackers not knowing seeds can distinguish a truly random number from a output of PRG with a negligible probability. Similar with PRG, hash function allows mapping arbitrary size of data to a fixed bit value. For reducing communication cost of FEVERLESS, we use `SHAKE-256` (Sha (2015)), one of the hash functions in `SHA-3` (Aumasson et al. (2008)) family, to generate customize size of maskings.

B.4 Key derivation function

Key Derivation Function (KDF) (Krawczyk & Eronen (2010)) is a kind of hash function that derives multiple secret keys from a main key by utilizing Pesudo-Random Function (PRF) (Kaliski (2005)). In general, KDF algorithm $DK \leftarrow KDF(mainkey, salt, rounds)$ derives keys DK based on a main key, a cryptographic salt and current round of processing algorithm. The security requires a secure KDF is robust for brute-force attack or dictionary attack. Inspired by (Zdziarski (2012)) where key shares generated by DH key exchange are converted to AES keys, in this paper, we use KDF to generate maskings for every round to reduce communication cost. The main key we use is generated by DH key exchange.

B.5 VERIFIABLE RANDOM FUNCTION

Verifiable Random Function (VRF) (Micali et al. (1999)) is a PRF providing verifiable proofs of correctness of outputs. It is a tool widely used in cryptocurrencies, smart contracts and leader selection in distributed systems (Micali (2016)). Basically, given a input x, a signature scheme and a hash function, a practical leader selection scheme with VRF (Micali (2016)) works as:

$$S_{leader} \leftarrow \mathsf{H}(\mathrm{sign}_{sk_i}(x)) \tag{6}$$

where sk_i is the secret key for i-th client, and the maximum leader score S_{leader} is used to determine leader. The security and unforgeability of VRF requires that the signature scheme has the property of uniqueness, and hash function is able to map the signature to a random string with fixed size. The correctness of this S_{leader} is proved by the signature of x.

B.6 DIFFERENTIAL PRIVACY

Differential Privacy (DP) (Dwork et al. (2006a;b)) is a data protection system targeting on publishing statistical information of datasets while keeping individual data private. The security of DP requires that adversaries cannot distinguish statistically change from two datasets where an arbitrary data point is different.

The most widely used DP mechanism is called (ϵ, δ)-DP requiring less noise injected than original proposed ϵ-DP but with the same privacy level. The formal definition is given as follows.

Definition 5. **((ϵ, δ) - Differential Privacy)** Given two real positive numbers (ϵ, δ) and a randomized algorithm $\mathcal{A}$: $\mathcal{D}^n \rightarrow \mathcal{Y}$, the algorithm $\mathcal{A}$ provides (ϵ, δ) - differential privacy if for all data sets D, $D^{'} \in \mathcal{D}^n$ differing in only one data sample, and all $\mathcal{S} \subseteq \mathcal{Y}$:

$$Pr[\mathcal{A}(D) \in \mathcal{S}] \leq exp(\epsilon) \cdot Pr[\mathcal{A}(D^{'}) \in \mathcal{S}] + \delta. \tag{7}$$

Note the noise $\mathcal{N} \sim N(0, \Delta^2\sigma^2)$ will be put into the output of the algorithm, where Δ is l_2 - norm sensitivity of D and $\sigma = \sqrt{2\ln(1.25/\delta)}$ (Abadi et al. (2016)).

C PRIVACY CONCERN

Since we assume feature names are not public information for all clients, and the values of features never leave from clients, the privacy issues are mainly incurred by the leakage of label information.

C.1 INFERENCE ATTACK

During training process, gradients and hessians are sent to source client for L_{split} computation. For classification task, the single gradient is in range $(-1, 0) \cup (0, 1)$ for binary classification. According to Eq.(2), a label can be inferred as 1 and 0 if the range is $(-1, 0)$ and $(0, 1)$, respectively. Besides, hessian illustrated in Eq.(3) can leak a prediction of the corresponding data sample. With training processing, the prediction is increasingly closer to a true label. The source client and outside attackers can infer the true label with high probability. Gradients and hessians cannot be transmitted in plaintext. We thus use secure aggregation scheme to protect them from inference attack.

C.2 DIFFERENTIAL ATTACK

Differential attack can happen anytime and many times during the calculation of gradients and hessians. Figure 5 describes an example of differential attack taking place in single node split. After sorting feature1, the semi-honest source client defines 2 split candidates and further computes $G_{\{2,5\}} = g_2 + g_5$ and $G_{\{1,2,3,5\}} = g_2 + g_5 + g_1 + g_3$ for the candidates 1 and 2, respectively. Since the source client holds label 2, even if $G_{\{2,5\}}$ is derived by secure aggregation, the g_5 still can be revealed by $G_{\{2,5\}} - g_2$.

Another example for differential attack is shown in Figure 6. Assume split candidate 1 is the one for splitting root node. In the current tree structure, source client may split right node by computing L_{split} of split candidate 2. In this case, $G_{\{1,3\}}$ should be aggregated by source client. And the g_5 can be revealed by $G_{\{1,2,3,5\}} - G_{\{1,3\}} - g_2$, where $G_{\{1,2,3,5\}}$ is computed in the previous node.

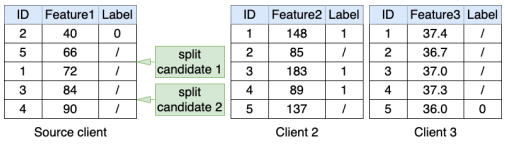

Figure 5: A differential attack on single node split

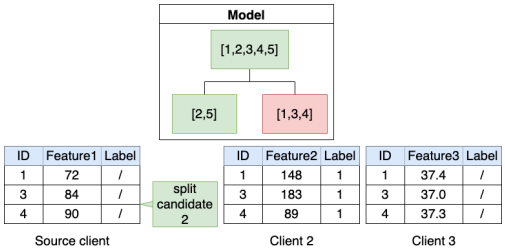

Figure 6: A differential attack on multiply node splits

D More details on FEVERLESS protocol

D.1 XGBoost Training over Distributed Labels

At the initial stage, we allow all clients to agree on a tree structure (maximum depth and the number of trees) and the learning rate for updating prediction. To avoid overfitting problem, we should define regularization parameters. Threshold impurity is also another vital parameter used to identify tree and leaf nodes via the maximum impurity. After that, we should choose ϵ, δ for DP, hash function for masking generation and noise leader selection. Besides, we select a multiplicative group $\mathbb{G}$ with order q generated by a generator g and a large prime number p to run DH.

At initialization process, all clients set parameters and sort their own feature based on values. Then, split candidates can be defined, and data samples between two different candidates will be grouped as a bucket. At the end, all entries are assigned initialized values to calculate the derivatives of loss function. The detailed algorithm is described as follows.

Algorithm 1: Initialization

Set parameters: all clients agree on the maximum depth of a tree d, the number of trees (NT), learning rate (η), regularization parameters (λ, γ), the threshold of L_{split}, ϵ, δ, p, g, and hash function
for $c \in [1, n]$ **do**
  for *each feature j owned by c* **do**
    $\texttt{sort}(X_j^{(c)})$
    define buckets: B_z^j
  end
  set initialized values: $\hat{y_i}^{(c)}$
end

After initialization, all clients can invoke Algorithm 2 to train model collaboratively. The inputs are from feature space consisting of features $X_j^{(c)}$ and labels $y_i^{(c)}$ distributed on different clients, respectively; while the output is a trained XGBoost model that can be used for prediction. Generally, trees are built one by one. And we see from line 4-10 in Algorithm 2 that each client can compute gradients and hessians at beginning of a new tree construction.

Following that, clients are to split current node. Note that XGBoost training in DL-VFL requires each client to calculate G and H. If the labels in some buckets are incomplete, the corresponding

gradients and hessians cannot be computed. Thus, each client should first broadcast missing data index set mID (see line 15-17 in Algorithm 2). Based on the predefined bucket B_z^j, mID can be defined if labels in B_z^j are not held by clients. In each broadcast, a client sending messages is regarded as a source client. Then others send the corresponding $g_i^{(c')}$ and $h_i^{(c')}$ back to the source client to compute L_{split} through Algorithm 3 depicted in Section D.2. After finding a maximum impurity $L_{split_max}^c$, the current node will be split to "left" and "right" nodes if $L_{split_max}^c > threshold_L_{split}$, in which the value of the split candidate is own by c.

In node splitting, clients should set a given node as "leaf" if current depth reaches the predefined maximum depth or the maximum L_{split} is less than the predefined threshold of L_{split} (see line 12, 24-32 in Algorithm 2). The derivation of leaf value is followed by Eq. 5 where G and H are intaken. Since a leaf node is either "left" or "right" split by one of the clients in $\mathcal{C}$ from its parent node, this client knows G and H and leaf value can be derived. Finally, this leaf value will be broadcast, and clients who own the corresponding $g_i^{(c)}$ and $h_i^{(c)}$ can use it to update predictions. The details for the above process are shown in Algorithm 2.

D.2 Secure Aggregation with Global Differential Privacy

In line 15-19 of Algorithm 2, source client is able to compute L_{split} from the requested missing data indexes and the aggregation of received messages. To avoid that inference and differential attacks are conducted on labels by source client and outside adversaries, we propose a privacy-preserving approach, shown in Algorithm 3, to "twist" the DH key exchange, noise leader selection and secure aggregation together. This method represents a viable alternative to train XGBoost securely in DL-VFL without demanding excessive computational resources and affecting model accuracy.

To generate the secure-but-can-be-cancelled-out maskings, we adopt DH here. In line 2-10 in Algorithm 3, all clients randomly select numbers as their secret keys and generate the corresponding public keys. For any two clients in the set $\mathcal{C}$, they will exchange public key and compute the corresponding shared keys. For simplicity, we do not describe the signature scheme for DH. We assume DH is conducted on authenticated channels, which means the man-in-the-middle attack (Khader & Lai (2015)) should be invalid here.

If the shared keys are used as maskings directly, our system is not robust for clients collusion unless the amount of communication has been sacrificed as a cost to update maskings per round. But the communication complexity is exponentially increased with the number of clients for a single node splitting. Considering the structure of trees, the overall communication complexity will be $O(2^d \cdot NT \cdot n^2)$, which may not scale well in practical applications.

To tackle this issue, we use KDF to update maskings per round automatically. Specifically, in line 24-25 of Algorithm 3, shared keys are taken as main keys. 0 and 1 are salt values for gradients and hessians, respectively. Since query in each round varies, the generated maskings should be dynamic accordingly. Besides, the sign of maskings is determined by the indexes of clients. In this way, we only need to use DH once, and the communication complexity is independent with tree structure.

To enable FEVERLESS to hold against differential attack, we use GDP approach allowing the chosen one to inject a global noise to aggregated values per round. The approach is quite subtle. If the noise leader is selected by source client, the system will be vulnerable to the collusion. Moreover, a client could be easily identified as a target if we choose it in advance, e.g., selecting a list of leaders before the training. To avoid these issues, we use VRF to iteratively select the leader (see line 12-21 in Algorithm 3). Specifically, in each round, the selection scores are generated by VRF based on secret key and the round counts. All clients can broadcast their scores and then the one who holds the "max value" will become the leader.

Before replying to source client, we have that the clients with labels put maskings to gradients and hessians, and for those without labels, they just generate and later send out maskings, in which the noise leader (i.e. one of the maskings generators) injects the noise. In this way, the maskings, guaranteeing perfect secrecy of the messages, will be cancelled out after the values aggregation, and the differentially private noise will solidate indistinguishability of individual data entry.

Algorithm 2: Protocol overview

Input: $\{X_j^{(c)} \mid j \in f, c \in |\mathcal{C}|\}$: features, $\{y_i^{(c)} \mid i \in m, c \in |\mathcal{C}|\}$: labels
Output: XGBoost model
Building trees:
for $nt \in [1, NT]$ **do**
  for $c \in [1, n]$ **do**
    for *each data entry* i *owned by* c **do**
      $g_i^{(c)} \leftarrow \partial_{\hat{y_i}^{(c)}} Loss(\hat{y_i}^{(c)}, y_i^{(c)})$
      $h_i^{(c)} \leftarrow \partial^2_{\hat{y_i}^{(c)}} Loss(\hat{y_i}^{(c)}, y_i^{(c)})$
    end
  end
  for *each node in the current tree* **do**
    while *current depth* $<d$ **do**
      for $c \in [1, n]$ **do**
        for *each feature j owned by c* **do**
          for *each* B_z^j *owned by c* **do**
            `Broadcast` $mID = \{i \mid y_i \notin \mathcal{Y}^c\}$
          end
          aggregate G, H by **Algorithm** 3
          compute L_{split} according to Eq.(4)
        end
        find the maximum $L_{split}^{(c)}$ and broadcast
      end
      $L_{split_max}^{(c)} \leftarrow \texttt{max}(\{L_{split}^{(c)} \mid c \in [1, n]\})$
      if $L_{split_max}^{(c)} \leq threshold_L_{split}$ **then**
        set current node as leaf node
        c computes w and broadcast
        `Break`
      else
        c splits current node to left node and right node, and broadcasts data index of them.
      end
    end
    set remaining nodes as leaf nodes
    c computes w and broadcast
    clients participating in calculation of w: update $\hat{y_i}^{(c)}$
  end
end

Note that in line 24-31 of Algorithm 3, the maskings and masked values are in the range $[0, N-1]$. And N should be sufficiently large to avoid overflow, and the summation of gradients and hessians should not exceed N.

Algorithm 3: Secure aggregation with global differential privacy

1 **Diffie-Hellman key exchange:**
2 **for** $c \in [1, n]$ **do**
3 $\quad sk_c \leftarrow \mathbb{Z}_p^*$
4 **end**
5 **for** $c \in [1, n]$ **do**
6 $\quad pk_c = g^{sk_c} \bmod p$
7 $\quad$ **for** $c^{'} \in [1, n] \wedge c^{'} \neq c$ **do**
8 $\quad\quad S_{c,c'} = pk_c^{sk_{c'}} \bmod p$
9 $\quad$ **end**
10 **end**
11 **Noise leader selection:**
12 `count` = 1
13 **for** *each time run this algorithm* **do**
14 $\quad$ **for** $c \in [1, n] \ \wedge \ c \neq source\ client$ **do**
15 $\quad\quad selec_c \leftarrow \mathsf{H}(\mathtt{SIGN}_{sk_c}(\mathtt{count}))$
16 $\quad\quad$ `Broadcast`
17 $\quad$ **end**
18 $\quad selec_c^{max} \leftarrow \mathtt{max}(\{selec_c \mid c \in [1, n]\})$
19 $\quad$ set c as noise leader
20 $\quad$ `count`+=1
21 **end**
22 **Secure aggregation:**
23 **for** $c \in [1, n]$ **do**
24 $\quad \mathtt{mask}_g^{(c)} \leftarrow \left(\sum_{c \neq c'} \frac{|c-c'|}{c-c'} \cdot \left(\mathsf{H}(S_{c,c'} \| 0 \| \mathtt{query}) \bmod N\right)\right) \bmod N$
25 $\quad \mathtt{mask}_h^{(c)} \leftarrow \left(\sum_{c \neq c'} \frac{|c-c'|}{c-c'} \cdot \left(\mathsf{H}(S_{c,c'} \| 1 \| \mathtt{query}) \bmod N\right)\right) \bmod N$
26 $\quad G^{(c)} = \sum_{i \in mIDs} g_i^{(c)} + \mathtt{mask}_g^{(c)} \bmod N,$
27 $\quad H^{(c)} = \sum_{i \in mIDs} h_i^{(c)} + \mathtt{mask}_h^{(c)} \bmod N$
28 $\quad$ **if** $c = leader$ **then**
29 $\quad\quad G^{(c)} = G^{(c)} + N(0, (\Delta_g \sigma)^2) \bmod N$
30 $\quad\quad H^{(c)} = H^{(c)} + N(0, (\Delta_h \sigma)^2) \bmod N$
31 $\quad$ **end**
32 $\quad$ send $\{G^{(c)}, H^{(c)}\}$ to source client
33 **end**

E SECURITY ANALYSIS

We investigate the security and privacy properties of our protocol. First, we define the security model of our setting and the properties. Then, we prove that our protocol satisfies these properties.

Security Model. Our security is based on the random oracle model (ROM) (Smart (2016)) where the hash function outputs uniformly random value for a new query and the same value for a previously answered query.

Adversarial Model. Our protocol is designed for *semi-honest security* model (Smart (2016)) where all parties follow the protocol while trying to obtain information regarding other parties' inputs. We assume that the source client can collude with other clients, but the size of colluding clients is no more than $n-2$.

E.1 PRIVACY GOALS

Our privacy goals can be summarized as:
• *Label privacy*: No adversary controlling at most $n-2$ clients can learn who is the owner of a label among the honest parties.
• *Data privacy*: No adversary controlling at most $n-2$ clients can extract the data of an honest party.

We first investigate the case where the source client is not part of the adversary. In the following theorem, we show that there exists a simulator Sim that simulates the joint view of clients in $\mathcal{A}$ by only using the inputs belonging to them. This implies that $\mathcal{A}$ does not learn more than what they have.

Theorem E.1 ($\mathcal{A}$ not including source client)**.** *There exists a* PPT *simulator* Sim *for all* $|\mathcal{C}| := n \geq 3$*,* $|\mathcal{X}| := f \geq n$*,* $|\mathcal{Y}| := m \geq 1$*,* $\bigcup_{c\in\mathcal{C}} \mathcal{X}^{(c)}$*,* $\bigcup_{c\in\mathcal{C}} \mathcal{Y}^{(c)}$ *and* $\mathcal{A} \subset \mathcal{C}$ *such that* $|\mathcal{A}| \leq n-2$*, the output of* Sim *is indistinguishable from the output of* $\texttt{REAL}$*:*

$$\texttt{REAL}_{\mathcal{A}}^{\mathcal{C},\mathcal{X},\mathcal{Y}}(\mathcal{X}^{\mathcal{C}}, \mathcal{Y}^{\mathcal{C}}) \equiv \mathsf{Sim}_{\mathcal{A}}^{\mathcal{C},\mathcal{X},\mathcal{Y}}(\mathcal{X}^{\mathcal{A}}, \mathcal{Y}^{\mathcal{A}}) \tag{8}$$

Proof. In order to prove that simulator Sim can simulate the outputs of the honest parties in $\mathcal{H} := \mathcal{C} - \mathcal{A}$, we show that the distribution of the inputs belonging to the rest of the network cannot be distinguished from a randomly generated data. In this way, the simulator can use any dummy values as inputs of the honest parties to simulate their outputs.

We will simulate the view of the $\mathcal{A}$ regarding the messages broadcast by the honest clients. A client c, first makes a key exchange with others, then after some internal operations, outputs $G^{(c)}$ and $H^{(c)}$ values. Let us investigate $G^{(c)}$ value, which is in the form of $\sum_{i\in mIDs} g_i^{(c)} + \texttt{mask}_g^{(c)}$, except for the noise leader who has additional noise of $N(0, (\Delta_g\sigma)^2)$. The mask values are computed as $\sum_{c\neq c'} \frac{|c-c'|}{c-c'} \cdot \mathsf{H}(S_{c,c'} \| 0 \| \texttt{query}) \bmod N$.

Here, we will use a hybrid model where we modify the protocol in several steps, and for each step, we will show that modifications are indistinguishable for the adversary $\mathcal{A}$. In the end, we will achieve a hybrid that can be simulated by Sim.

$\texttt{Hybrid}_1$: The first hybrid directly follows the protocol. The distribution of the variables and the view of $\mathcal{A}$ is the same as $\texttt{REAL}$.

$\texttt{Hybrid}_2$: In the second hybrid, we replace the agreed keys between honest clients $S_{c,c'}$ for all $c, c' \in \mathcal{H}$ with random values $r_{c,c'} \in G$ where G is the group of key exchange protocol G. In the original protocol, Diffie-Hellman key exchange is used. The replacement is indistinguishable for the adversary because of the decision Diffie-Hellman assumption given in Definition 4.

Also, note that these random values are only available to parties involved in the key exchange unless they are corrupted by the adversary.

$\texttt{Hybrid}_3$: In this hybrid, we replace the mask values of honest clients $\texttt{mask}_g^{(c)}$ for all $c \in \mathcal{H}$ with random values $R^{(c)}$. Note that with the replacement in the previous step, the mask values are

computed via $\sum_{c \neq c'} \frac{|c-c'|}{c-c'} \cdot \mathsf{H}(r_{c,c'} \| 0 \| \texttt{query}) \bmod N$ where $r_{c,c'} \in \mathcal{Z}_N$ is a random value that is unknown to the adversary (if both c and c' are honest). Because of the random oracle model, the output of the hash function will be a uniformly random value that is also unknown to the adversary. Since there are at most $n-2$ clients in $\mathcal{A}$, we have at least two honest clients c and c' for which the adversary cannot know the uniformly chosen output of $\mathsf{H}(r_{c,c'} \| 0 \| \texttt{query})$. Then, the modular summation of these outputs includes at least one value that the adversary does not know and is uniformly random. Thus, it cannot be distinguishable from a random value $R^{(c)}$.

Hybrid$_4$: In this hybrid, we replace gradients of honest clients $g_i^{(c)}$ for all $c \in \mathcal{H}$ with '0's. This is done by replacing mask values with $\overline{R^{(c)}} := R^{(c)} - \sum_{i \in mIDs} g_i^{(c)} \bmod N$ to keep the $G^{(c)}$ value the same. From the adversary's perspective, since $R^{(c)}$ values are unknown and uniformly randomly chosen, the replacement is not distinguishable.

In **Hybrid**$_4$, we replace the gradients of honest parties with '0's, and the mask values are replaced by $\overline{R^{(c)}}$ which is unknown to the adversary and chosen from a uniform distribution. Thus, a simulator Sim can simulate the outputs of honest parties $G^{(c)}$ without necessarily knowing their inputs.

The same can be analyzed for hessian value, $H^{(c)}$. Since the masking values of $G^{(c)}$ and $H^{(c)}$ are different and the hash function is modeled as a random oracle, the randomness in both parts of them are independent of each other and indistinguishable by the adversary $\mathcal{A}$. Overall, the simulator Sim can simulate our protocol.

Thus, the view of the $\mathcal{A}$ can be simulated by replacing the inputs of the honest parties with zeros. Thus, the adversary does not learn any information on the inputs of the honest parties. □

Now, we analyze the case where the source client is part of the $\mathcal{A}$. We show that there exists a simulator Sim that simulates the joint view of clients in $\mathcal{A}$ by only using the inputs belonging to them and the summations G and H. This implies that $\mathcal{A}$ does not learn more than what they have and the summation.

Theorem E.2 ($\mathcal{A}$ including source client)**.** *There exists a* PPT *simulator* Sim *for all* $|\mathcal{C}| := n \geq 3$*,* $|\mathcal{X}| := f \geq n$*,* $|\mathcal{Y}| := m \geq 1$*,* $\bigcup_{c \in \mathcal{C}} \mathcal{X}^{(c)}$*,* $\bigcup_{c \in \mathcal{C}} \mathcal{Y}^{(c)}$ *and* $\mathcal{A} \subset \mathcal{C}$ *such that* $|\mathcal{A}| \leq n-2$*, the output of* Sim *is indistinguishable from the output of* $\texttt{REAL}$*:*

$$\texttt{REAL}_{\mathcal{A}}^{\mathcal{C},\mathcal{X},\mathcal{Y}}(\mathcal{X}^{\mathcal{C}}, \mathcal{Y}^{\mathcal{C}}) \equiv \mathsf{Sim}_{\mathcal{A}}^{\mathcal{C},\mathcal{X},\mathcal{Y}}(G, H, \mathcal{X}^{\mathcal{A}}, \mathcal{Y}^{\mathcal{A}}) \tag{9}$$

where

$$G = \sum_{i \in mIDs} g_i^{(c)} + N(0, (\Delta_g \sigma)^2), H = \sum_{i \in mIDs} h_i^{(c)} + N(0, (\Delta_h \sigma)^2).$$

Proof. Here, we again show that Sim can simulate the outputs of the honest parties in $\mathcal{H}$ without knowing their inputs. Unlike Theorem E.1, Sim is also given the summations G and H because the adversary includes the source client.

We can use the same hybrids with Theorem E.1 until **Hybrid**$_4$, this is because that the inputs of the honest clients are not required yet. We need to update **Hybrid**$_4$ such that it takes into account the summation. Here are the hybrids for the $\mathcal{A}$ with source client:

Hybrid$_1$,**Hybrid**$_2$,**Hybrid**$_3$: The same with Theorem E.1.

Hybrid$_4$: In this hybrid, we replace gradients of honest clients $g_i^{(c)}$ for all $c \in \mathcal{H}$ with '0's, except one c' which will be equal to $\sum_{i \in mIDs} g_i^{(\mathcal{H})} \bmod N = G - \sum_{i \in mIDs} g_i^{(\mathcal{A})} \bmod N$. The honest client c' is randomly chosen among $\mathcal{H}$. From the adversary's perspective, since $R^{(c)}$ are unknown uniformly random chosen values, the replacement is not distinguishable.

Overall, the view of the $\mathcal{A}$ can be simulated by replacing the inputs of the honest parties with zeros, except one with $\sum_{i \in mIDs} g_i^{(\mathcal{H})} \bmod N$. Thus, $\mathcal{A}$ does not learn any information from the honest clients, except the summation $\sum_{i \in mIDs} g_i^{(\mathcal{H})} \bmod N$. □

With Theorem E.2, we show that even the adversary $\mathcal{A}$ with source client cannot know more than the summation of gradient and hessian values, G and H. The proof is done via Sim without requiring

individual data of the honest clients except for the summation. This implies that the adversary cannot distinguish which party provided which gradient or hessian values. Moreover, the parties who do not have any of the requested g or h values will send '0' together with the mask (and noise for the leader). This implies that we provide *label privacy*. Meaning that the adversary cannot distinguish which label's g or h values are coming from which honest client.

In the case when the adversary includes the source client, the summation of gradient and hessian values can be known to the adversary. In the following theorem, we show that these summations do not leak any individual data due to differential privacy.

Theorem E.3 (Privacy of the Inputs)**.** *No $\mathcal{A} \subset \mathcal{C}$ such that $|\mathcal{A}| \leq n-2$ can retrieve the individual values of the honest clients.*

Proof. If the adversary does not include the source client, then following the previous theorems, the adversary cannot know any of the inputs belonging to the honest parties. Otherwise, it knows the summations G and H. Since we apply differential privacy Dwork et al. (2006b;a), the summation cannot leak information regarding the inputs. According to Definition 5, we add differentially private noise guaranteeing the security of individual data points while summation can be calculated. Note that because of the secure aggregation, the adversary cannot learn anything but the summation. Thus, our protocol does not require the addition of noise to each data. Instead, we only require the noise leader to add the noise, which prevents the retrieval of the individual data from the summation. □

In Theorems E.1 and E.2, we show that $\mathcal{A}$ cannot distinguish the individual values from randomly chosen values and can only know the summation if the source is part of the adversary. In Theorem E.3, we show that $\mathcal{A}$ cannot extract the individual values of the users from the summation due to the added noise and differential privacy. Thus, our protocol satisfies *data privacy*. In other words, the adversary cannot learn the data point of an honest client.

It is important to note that since the noise leader is selected via VRF, no adversary can guess if any honest party will be the leader in the upcoming round beforehand. This provides additional security regarding the manipulation of the noise leader.

F Discussion

Hiding labels distribution. In the proposed protocol, source client broadcasts the missing data indexes mID (line 16 of Algorithm 2). Under the semi-honest setting, if source client sends missing indexes consistently, the adversaries will figure out which labels are distributed on source clients by statistic analysis. We note that FEVERLESS can be expanded to avoid this type of leakage by yielding extra communication overheads. Specifically, during broadcasting period, source client should send indexes of one bucket instead of mID, and the rest of protocol remains constant. In this way, others cannot distinguish the distribution of labels because all clients share the same index set $\mathcal{I}$. If we assume labels are uniformly distributed on each client, the extra overheads are restricted to $|\mathcal{I}|/|\mathcal{C}|$. This cost is clearly noticeable in those datasets with a large number of data points.

Other security tools. The masking scheme realizing secure aggregation may be replaced with an MPC (Damgård et al. (2012); Wu et al. (2020)) or additively homomorphic encryption (Paillier (1999)). However, the major defect of these tools is that they entail labor-intensive calculation with regard to encryption, which may not scale well in large-scale datasets. Due to this concern, we only put light-weight computation in FEVERLESE and further, we enhance the security to "perfect secrecy".

In our design, the selection of noise leader is captured by VRF. We note that there may be other options to fulfil the goal. For example, Proof of Elapsed Time (PoET) (Chen et al. (2017); Corso (2019)) is an interesting and effective mechanism which is used to maintain the consensus of distributed peers in Hyperledger Sawtooth. It provides a fair and trusted lottery strategy to select a block winner (per consensus round). Sharing the same philosophy with the VRF, it may be deployed in our protocol to yield leader. And building a more efficient noise leader selection algorithm could be an interesting open problem.

G MORE DETAILS ON DATESETS

• **Credit Card:** It is a commercial dataset used for predicting whether costumers will make payment on time. It provides 30,000 samples, and each sample composes of 23 features.
• **Bank marketing:** Consisting with 45,211 data points and 17 features, the goal of bank marketing is to predict if a client will subscribe a term deposit.
• **Banknote authentication:** Offering 1,372 data points and 4 features, this dataset is used to classify authenticated and unauthenticated banknotes. Note that different from traditional tabular data, features in the dataset are extracted from images that are taken from genuine and forged banknote-like specimens through Wavelet Transform (Antonini et al. (1992)). Using the small-scale dataset, the trained model may not be robust for noise, which brings negative impact on accuracy.

H ADDITIONAL EXPERIMENTS AND FIGURES

We present additional experiments, and all the experimental settings follow those defined in Section 4.1. In each presented figure, we show the results executed on the datesets Credit card (left), Bank Marketing (middle) and Banknote Authentication (right). Note that the comparison among FEVERLESS, LDP, and AHE requires a condition that #client=2; when #client=1, we can only show the results of the baseline. And the average performance of FEVERLESS in these figures is highlighted as the red dotted line.

Figures 7-18 elaborate that how accuracy varies with the increase of client number among the baseline, FEVERLESS and LDP, w.r.t. different tree structure and ϵ. In general, without any added noise, the baseline can reach the highest accuracy and meanwhile, the accuracy remains stable as the client number increases. The performance of FEVERLESS is right behind that of baseline but still keeps stable. Note there are slight fluctuations in some figures (e.g. Figure 10, 12 and 14), especially for the case where complex tree structure and small ϵ are used. The LDP approach does harm accuracy, which can be seen from the continuously and significantly falling bars in the figures. Naturally, when more clients engage into the training, more noise should be added into the model. This makes LDP's performance far lower than the red line.

Note that banknote dataset is composed of 4 features. In the VFL setting, every client should have at least one feature. Therefore, we can only allow up to 4 clients to participate in the training. Beside, FEVERLESS does not perform well in banknote dataset. This is so because the model is trained by a small number of samples, so that the robustness is seriously affected by noise.

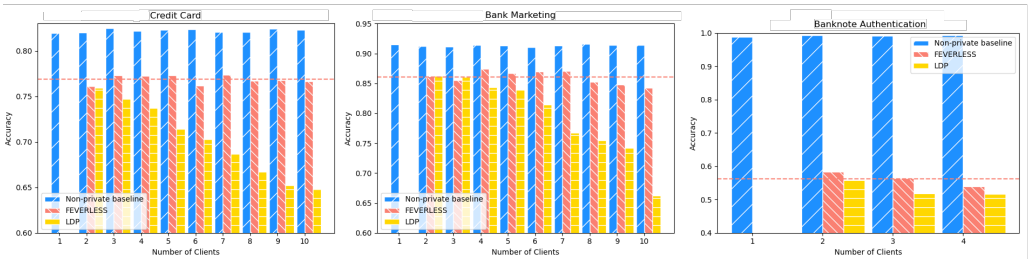

Figure 7: Comparison of accuracy in depth:10, the number of trees:10, epsilon:10.

In Figures 19-24, we show the time performance based on various numbers of client, tree and depth. In general, the runtime of FEVERLESS is slightly higher that that of the baseline. Compared to AHE, FEVERLESS significantly reduces training time while preserving privacy. This advantage is clearly seen from the cases using complex tree structures. Note that AHE can be replaced by other more complex cartographic solutions, such as secure MPC, which can also maintain data/label privacy. But these MPC-based solutions will consume more runtime.

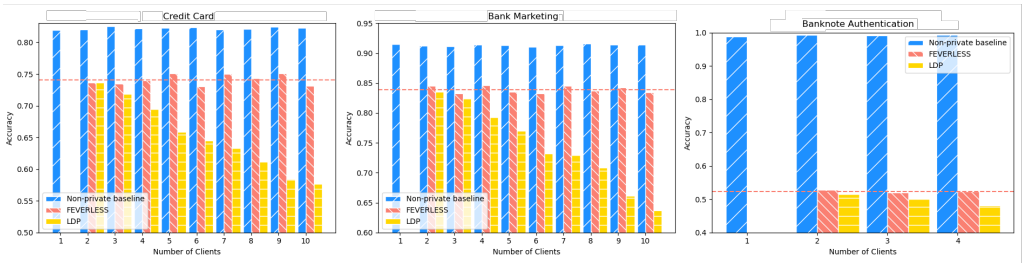

Figure 8: Comparison of accuracy in depth:10, the number of trees:10, epsilon:5.

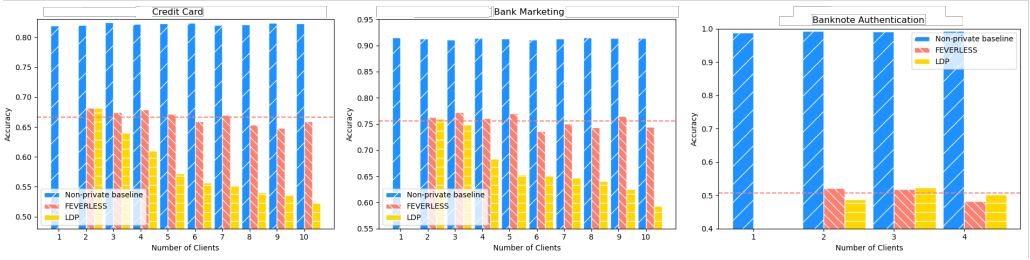

Figure 9: Comparison of accuracy in depth:10, the number of trees:10, epsilon:2.

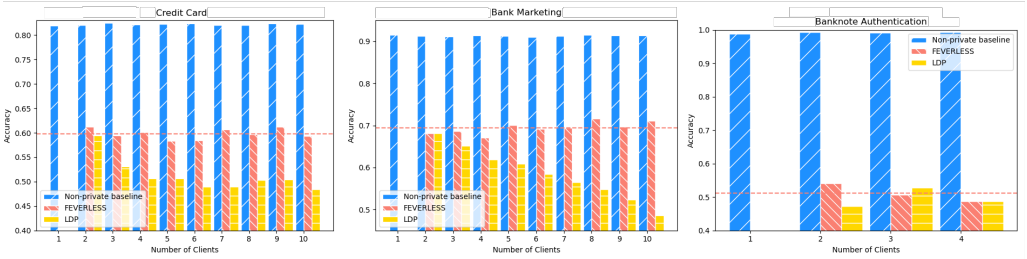

Figure 10: Comparison of accuracy in depth:10, the number of trees:10, epsilon:1.

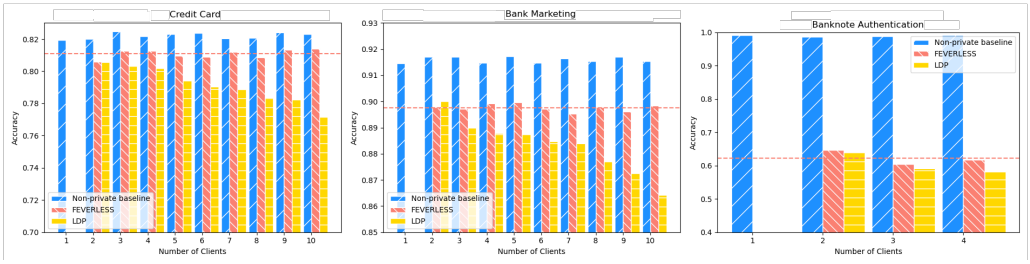

Figure 11: Comparison of accuracy in depth:8, the number of trees:8, epsilon:10.

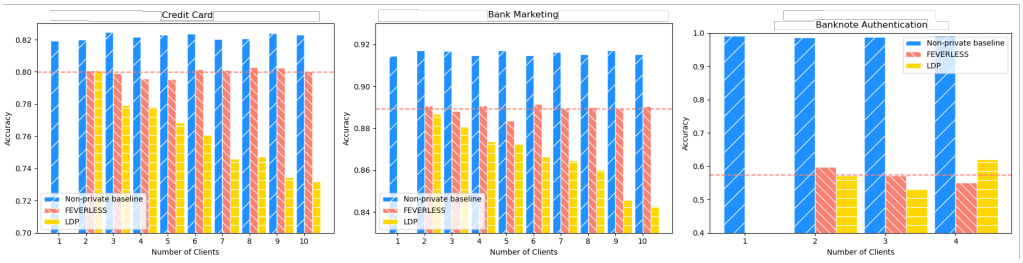

Figure 12: Comparison of accuracy in depth:8, the number of trees:8, epsilon:5.

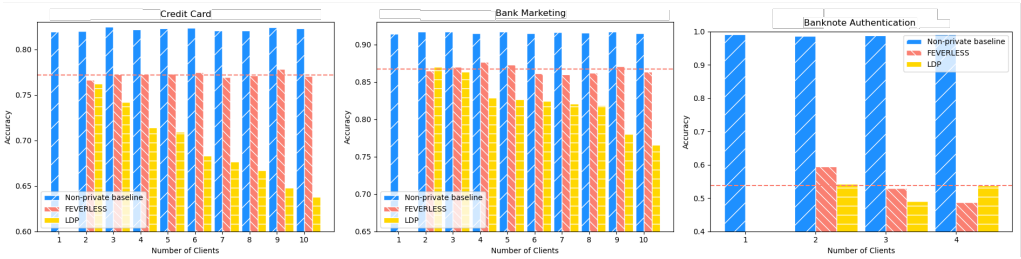

Figure 13: Comparison of accuracy in depth:8, the number of trees:8, epsilon:2.

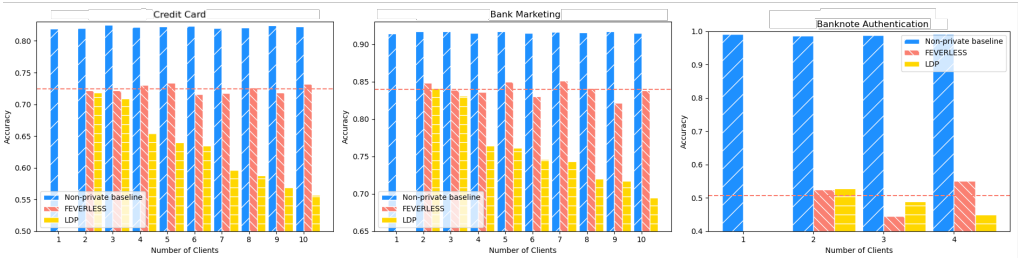

Figure 14: Comparison of accuracy in depth:8, the number of trees:8, epsilon:1.

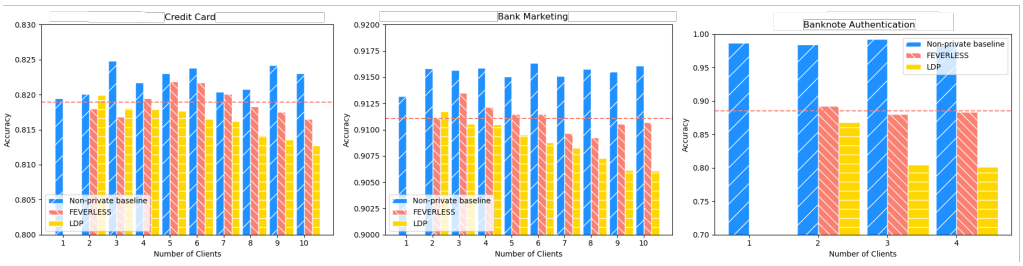

Figure 15: Comparison of accuracy in depth:6, the number of trees:6, epsilon:10.

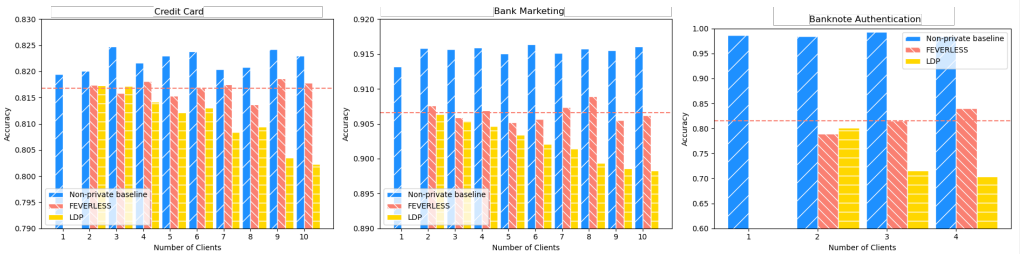

Figure 16: Comparison of accuracy in depth:6, the number of trees:6, epsilon:5.

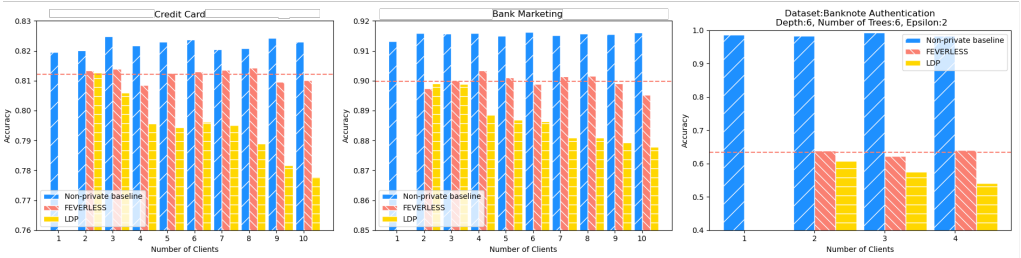

Figure 17: Comparison of accuracy in depth:6, the number of trees:6, epsilon:2.

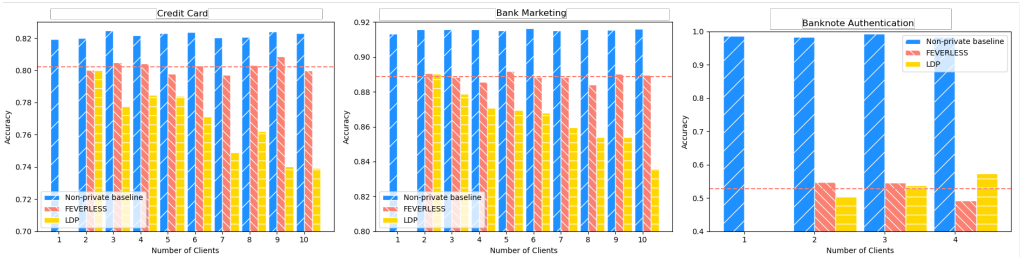

Figure 18: Comparison of accuracy in depth:6, the number of trees:6, epsilon:1.

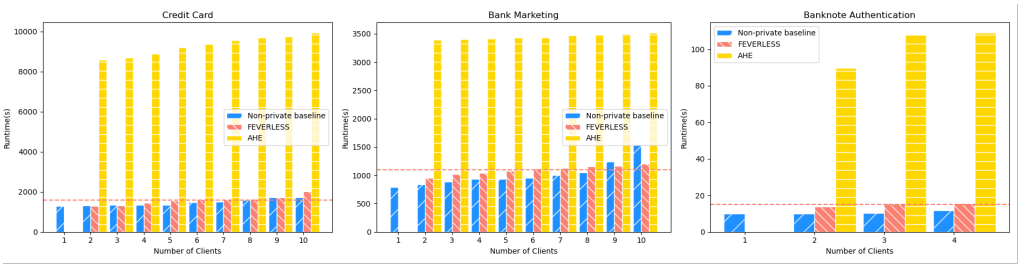

Figure 19: Comparison of runtime in depth:10, the number of trees:10.

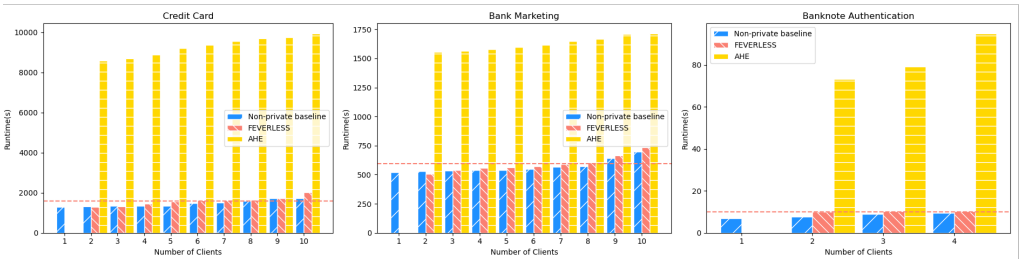

Figure 20: Comparison of runtime in depth:8, the number of trees:8.

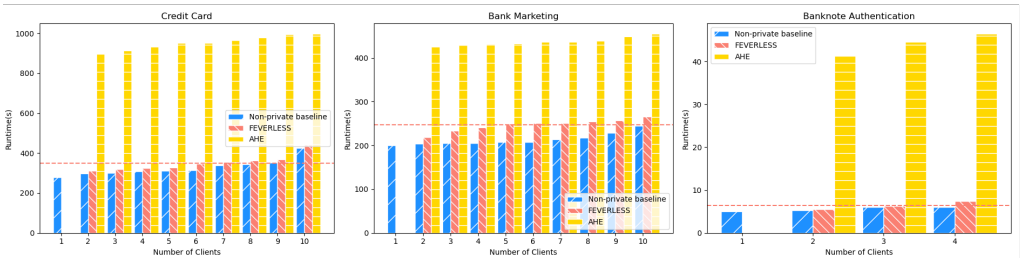

Figure 21: Comparison of runtime in depth:6, the number of trees:6.

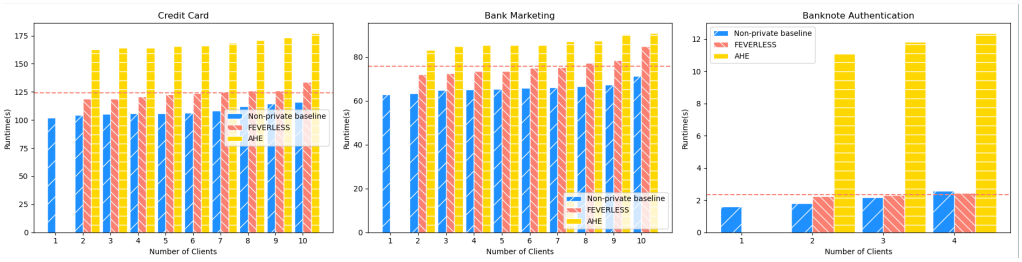

Figure 22: Comparison of runtime in depth:4, the number of trees:4.

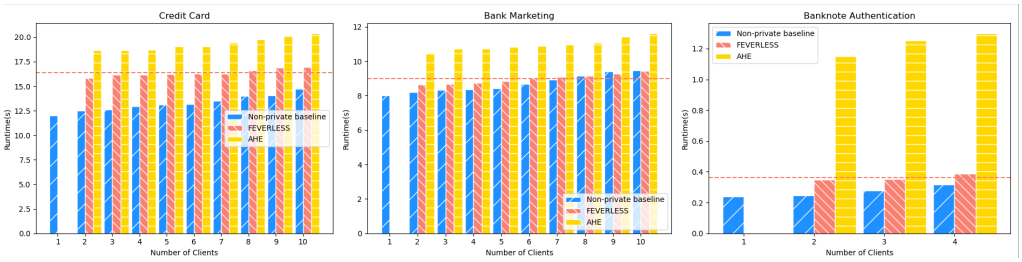

Figure 23: Comparison of runtime in depth:2, the number of trees:2.