SecureBoost: A Lossless Federated Learning Framework

Kewei Cheng, *University of California, Los Angeles, Los Angeles, 90095, USA*

Tao Fan, *WeBank, Shenzhen, 518052, China*

Yilun Jin, *Hong Kong University of Science and Technology, Hong Kong*

Yang Liu and Tianjian Chen, *WeBank, Shenzhen, 518052, China*

Dimitrios Papadopoulos, *Hong Kong University of Science and Technology, Hong Kong*

Qiang Yang, *WeBank, Shenzhen, 518052, China*

The protection of user privacy is an important concern in machine learning, as evidenced by the rolling out of the General Data Protection Regulation (GDPR) in the European Union (EU) in May 2018. The GDPR is designed to give users more control over their personal data, which motivates us to explore machine learning frameworks for data sharing that do not violate user privacy. To meet this goal, in this article, we propose a novel lossless privacy-preserving tree-boosting system known as SecureBoost in the setting of federated learning. SecureBoost first conducts entity alignment under a privacy-preserving protocol and then constructs boosting trees across multiple parties with a carefully designed encryption strategy. This federated learning system allows the learning process to be jointly conducted over multiple parties with common user samples but different feature sets, which corresponds to a vertically partitioned dataset. An advantage of SecureBoost is that it provides the same level of accuracy as the non -privacy-preserving approach while at the same time, reveals no information of each private data provider. We show that the SecureBoost framework is as accurate as other nonfederated gradient tree-boosting algorithms that require centralized data, and thus, it is highly scalable and practical for industrial applications such as credit risk analysis. To this end, we discuss information leakage during the protocol execution and propose ways to provably reduce it.

The modern society is increasingly concerned with the unlawful use and exploitation of personal data. At the individual level, improper use of personal data may cause potential risk to user privacy. At the enterprise level, data leakage may have grave consequences on commercial interests. Actions are being taken by different societies. For example, the European Union has enacted a law known as General Data Protection Regulation (GDPR). GDPR is designed to give users more control over their personal data.[1–4] Many enterprise that rely heavily on machine learning are beginning to make sweeping changes as a consequence.

1541-1672 © 2021 IEEE
Digital Object Identifier 10.1109/MIS.2021.3082561
Date of publication 25 May 2021; date of current version 17 December 2021.

Despite difficulty in meeting the goal of user-privacy protection, the need for different organizations to collaborate while building machine learning models still stays strong. In reality, many data owners do not have sufficient amount of data to build high-quality models. For example, retail companies have users' purchases and transaction data, which are highly useful if provided to banks for credit rating applications. Likewise, mobile phone companies have users' usage data, but each company may only have a small amount of users, which are not enough to train high-quality user preference models. Such companies have strong motivation to collaboratively exploit the joint data value.

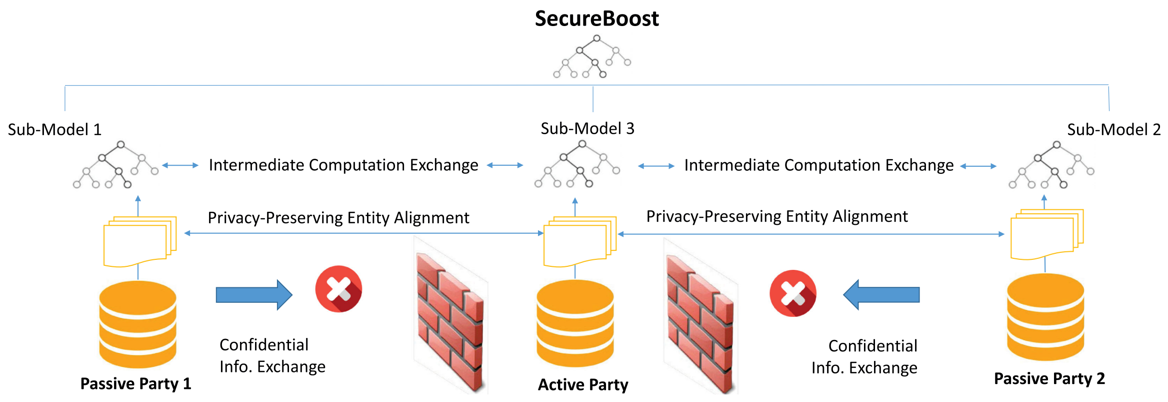

SecureBoost
Sub-Model 1
Sub-Model 3
Sub-Model 2
Intermediate Computation Exchange
Intermediate Computation Exchange
Privacy-Preserving Entity Alignment
Privacy-Preserving Entity Alignment
Confidential Info. Exchange
Confidential Info. Exchange
Passive Party 1
Active Party
Passive Party 2

FIGURE 1. Illustration of the proposed SecureBoost framework.

So far, it is still a challenge to allow different data owners to collaboratively build high-quality machine learning models while at the same time protecting user-data privacy and confidentiality. In the past, several attempts have been made to address the user-privacy issue in machine learning.[5,6] For example, Apple proposed to use *differential privacy* (DP)[7,8] to address the privacy preservation issue. The basic idea of DP is to add properly calibrated noise to data to disambiguate the identity of any individuals when data are being exchanged and analyzed by a third party. However, DP only prevents user-data leakage to a certain degree and cannot completely rule out the identity of an individual. In addition, data exchange under DP still requires that data change hands between organizations, which may not be allowed by strict laws like GDPR. Furthermore, the DP method is *lossy* in machine learning in that models built after noise is injected may perform unsatisfactorily in prediction accuracy.

More recently, Google introduced a *federated learning* (FL) framework[9] and deployed it on Android cloud. The basic idea is to allow individual clients to upload only model updates but not raw data to a central server where the models are aggregated. A secure aggregation protocol was further introduced[10] to ensure that the model parameters do not leak user information to the server. This framework is also referred to as horizontal FL[11] or data-partition FL where each partition corresponds to a subset of data samples collected from one or multiple users.

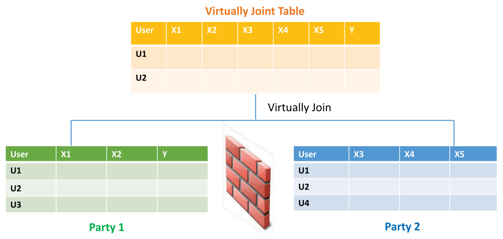


FIGURE 2. Vertically partitioned dataset.

In this article, we consider another setting where multiple parties collaboratively build their machine learning models while protecting user privacy and data confidentiality. Our setting is shown in Figure 2 and is typically referred as vertical FL[11] because data are partitioned by features among different parties. This setting has a wide range of real-world applications. For example, financial institutes can leverage alternative data from a third party to enhance users' and small- and medium- enterprises' credit ratings.[12] Patents' record from multiple hospitals can be used together for diagnoses.[13,14] We can regard the data located at different parties as a subsection of a virtual big data table obtained by taking the union of all data at different parties. Then, the data at each party have the following properties.

1) The big data table is vertically split, such that the data are split in the feature dimension among parties.
2) Only one data provider has the label information.
3) Parties share a common set of users.

Our goal is then to allow parties to build a prediction model jointly while protecting all parties from leaking data information to other parties. In contrast with most existing work on privacy-preserving data mining and machine learning, the complexity in our setting is significantly increased. Unlike sample-partitioned/horizontal FL, the vertical FL setting requires a more complex mechanism to decompose the loss function at each party.[5,15,16] In addition, since only one data provider owns the label information, we need to propose a secure protocol to guide the learning process instead of sharing label information explicitly among all parties. Finally, data confidentiality and privacy concerns prevent parties from exposing their own users. Hence, entity alignment should also be conducted in a sufficiently secure manner.

Tree boosting is a highly effective and widely used machine learning method, which excels in many machine learning tasks due to its high efficiency as well as strong interpretability. For example, XGBoost[30] has been widely used in various applications including credit risk analysis and user behavior studies. In this article, we propose a novel end-to-end privacy-preserving tree-boosting algorithm and framework known as SecureBoost to enable machine learning in a federated setting. SecureBoost has been implemented in an open-sourced FL project, FATE,[a] to enable industrial applications. Our FL framework operates in two steps. First, we find the common users among the parties under a privacy-preserving constraint. Then, we collaboratively learn a shared classification or regression model without leaking any user information to each other. We summarize our main contributions as follows.

- We formally define a novel problem of privacy-preserving machine learning over vertically partitioned data in the setting of FL.
- We present an approach to train a high-quality tree boosting model collaboratively while keeping the training data local over multiple parties. Our protocol does not need the participation of a trusted third party.
- Finally and importantly, we prove that our approach is *lossless* in the sense that it is as accurate as any centralized non-privacy-preserving methods that bring all data to a central location.
- In addition, along with a proof of security, we discuss what would be required to make the protocols completely secure.

PRELIMINARIES AND RELATED WORK

To protect the privacy of the data used for learning a model, Shokri and Shmatikov[17] proposed to take advantage of DP for learning a deep learning model. Recently, Google introduced an FL framework to prevent the data from being transmitted by bringing the model training to each mobile terminal.[9,10,18] Its basic idea is that each local mobile terminal trains the local model using its local data with the same model architecture. The global model can simply be updated by averaging all the local models. Following the same idea, several attempts have been made to reinvent different machine learning models to the federated setting, including decision tree,[19,20] linear/logistic regression,[21–23] and neural network.[24,25]

All the above methods are designed for horizontally partitioned data. Unlike sample-partitioned/horizontal FL, the vertical FL setting requires a more complex mechanism to decompose the loss function at each party. The concept of vertical FL is first proposed in the works of Hardy *et al.*[5] and Yang *et al.*[11] and protocols are proposed for linear models[5,13] and neural networks.[26] Some previous works have been proposed for privacy-preserving decision trees over vertically partitioned data.[16,27] However, their proposed methods have to reveal class distribution over given attributes, which will cause potential security risks. In addition, they can only handle discrete data, which is less practical for real-life scenarios. In contrast, our method guarantees better protection to the data and can be easily applied to continuous data. Another work proposed by Djatmiko *et al.*[28] jointly performs logistic regression over the encrypted vertically partitioned data by approximating a nonlinear logistic loss by a Taylor expansion, which will inevitably compromise the performance of the model. In contrast to these works, we propose a novel approach that is *lossless* in nature.

PROBLEM STATEMENT

Let $\{\mathbf{X}^k \in \mathbb{R}^{n_k \times d_k}\}_{k=1}^{m}$ be the data matrix distributed on m private parties with each row $\mathbf{X}^k_{i*} \in \mathbb{R}^{1\times d_k}$ being a data instance. We use $\mathcal{F}^k = \{f_1, \ldots, f_{d_k}\}$ to denote the feature set of corresponding data matrix $\mathbf{X}^k$. Two parties p and q have different sets of features, denoted as $\mathcal{F}^p \cap \mathcal{F}^q = \varnothing, \forall p \neq q \in \{1, \ldots, m\}$. Different parties may hold different sets of users as well, allowing some degree of overlap. Only one of the parties holds the class labels $\mathbf{y}$.

Definition 1. **Active Party:**

We define the active party as the data provider who holds both a data matrix and the class label. Since the class label information is indispensable for supervised learning, the active party naturally takes the responsibility as a dominating *server* in FL.

Definition 2. **Passive Party:**

We define the data provider that has only the data matrix as a passive party. Passive parties play the role of clients in the FL setting.

The problem of privacy-preserving machine learning over vertically split data in FL can be stated as follows:

[a]https://github.com/FederatedAI/FATE

Given: A vertically partitioned data matrix $\{\mathbf{X}^k\}_{k=1}^m$ distributed on m private parties and the class labels $\mathbf{y}$ distributed on active party.

Learn: A machine learning model M without giving information of the data matrix of any party to others in the process. Model M is a function that has a projection M_i at each party i, such that M_i takes input of its own features X_i.

Lossless Constraint: We require that model M is lossless, which means that the loss of M under FL over the training data is the same as the loss of M' when M' is built on the union of all data.

FL WITH SECUREBOOST

As one of the most popular machine learning algorithms, the gradient-tree boosting excels in many machine learning tasks, such as fraud detection, feature selection, and product recommendation. In this section, we propose a novel gradient-tree boosting algorithm called SecureBoost in the FL setting. It consists of two major steps. First, it aligns the data under the privacy constraint. Second, it collaboratively learns a shared gradient-tree boosting model while keeping all the training data secret over multiple private parties. We explain each step below.

Our first goal is to find a common set of data samples at all participating parties so as to build a joint model M. When the data are vertically partitioned among parties, different parties hold different but partially overlapping users, which can be identified using their IDs. The problem is how to find the common data samples across the parties without revealing the nonshared parts. To achieve this goal, we align the data samples under a privacy-preserving protocol for interdatabase intersections.[29]

After aligning the data across different parties under the privacy constraint, we now consider the problem of jointly building tree ensemble models over multiple parties without violating privacy. Before further discussing the details of the algorithm, we first introduce the general framework of FL. In FL, a typical iteration consists of four steps. First, each client downloads the current global model from server. Second, each client computes an updated model based on its local data and the current global model, which resides within the active party. Third, each client sends the model update back to the server under encryption. Finally, the server aggregates these model updates and construct the updated global model.

Following the general framework of FL, we see that to design a privacy-preserving tree-boosting framework in the setting of FL, essentially we have to answer the following three questions: 1) How can each client (i.e., a passive party) compute an updated model based on its local data without reference to class label? 2) How can the server (i.e., the active party) aggregate all the updated model and obtain a new global model? 3) How to share the updated global model among all parties without leaking any information at inference time? To answer these three questions, we start by reviewing a tree ensemble model, XGBoost,[30] in a nonfederated setting.

Given a dataset $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n samples and d features, XGBoost predicts the output by using K regression trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(\mathbf{x}_i). \tag{1}$$

To learn the set of regression tree models used in (1), it greedily adds a tree f_t at the tth iteration to minimize the following loss:

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

where $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \|w\|^2$, $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$.

When constructing the regression tree in the tth iteration, it starts from the tree with a depth of 0 and add a split for each leaf node until reaching the maximum depth. In particular, it maximizes the following equation to determine the best split:

$$\mathcal{L}_{sp} = \frac{1}{2} \left[\frac{\left(\sum_{i \in I_L} g_i\right)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{\left(\sum_{i \in I_R} g_i\right)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{\left(\sum_{i \in I} g_i\right)^2}{\sum_{i \in I} h_i + \lambda} \right] - \gamma \tag{3}$$

where I_L and I_R are the instance spaces of left and right tree nodes after the split.

After it obtains an optimal tree structure, the optimal weight w_j^* of leaf j can be computed by the following equation where I_j is the instance space of leaf j.

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

From the above review, we make following observations.

1) The evaluation of split candidates and the calculation of the optimal weight of leaf only depends on g_i and h_i, which makes it easily adapted to the setting of FL.

2) The class label can be inferred from g_i and h_i. For instance, when we take the square loss as the loss function, we have $g_i = \hat{y}_i^{(t-1)} - y_i$.

With the above observations, we now introduce our federated gradient tree boosting algorithm. Following observation 1), we can see that passive parties can determine their locally optimal split with only its local data and g_i, h_i, which motivates us to follow such method to decompose learning task at each party. However, according to observation 2), g_i and h_i should be regarded as sensitive data, since they are able to disclose class label information to passive parties. Therefore, in order to keep g_i and h_i confidential, the active party is required to encrypt g_i and h_i before sending them to passive parties. The remaining challenge is how to determine the locally optimal split with encrypted g_i and h_i for each passive party.

According to (3), the optimal split can be found if $g_l = \sum_{i \in I_L} g_i$ and $h_l = \sum_{i \in I_L} h_i$ are calculated for every possible split. So next, we show how to obtain g_l and h_l with encrypted g_i and h_i using an additive homomorphic encryption scheme.[31] The Paillier encryption scheme is taken as our encryption scheme.

Algorithm 1. Aggregate Encrypted Gradient Statistics

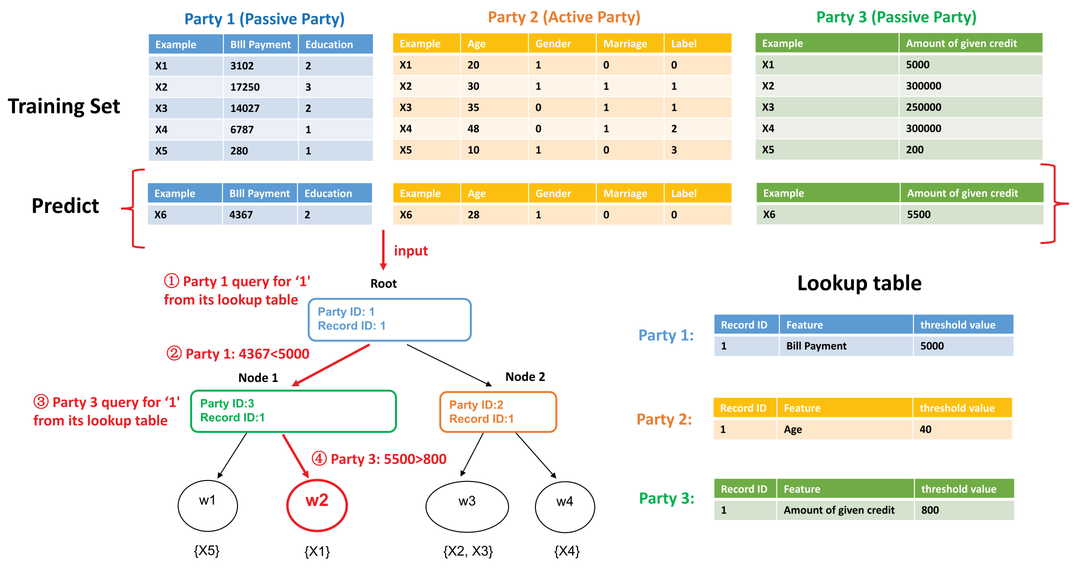
```
Input: $I$, instance space of current node
Input: $d$, feature dimension
Input: $\{\langle g_i \rangle, \langle h_i \rangle\}_{i \in I}$
Output: $\mathbf{G} \in \mathbb{R}^{d \times l}$, $\mathbf{H} \in \mathbb{R}^{d \times l}$
1:  for $k = 0 \rightarrow d$ do
2:      Propose $S_k = \{s_{k1}, s_{k2}, \ldots, s_{kl}\}$ by percentiles on
        feature $k$
3:  end for
4:  for $k = 0 \rightarrow d$ do
5:      $\mathbf{G}_{kv} = \sum_{i \in \{i | s_{k,v} \geq x_{i,k} > s_{k,v-1}\}} \langle g_i \rangle$
6:      $\mathbf{H}_{kv} = \sum_{i \in \{i | s_{k,v} \geq x_{i,k} > s_{k,v-1}\}} \langle h_i \rangle$
7:  end for
```

Denoting the encryption of a number u under the Paillier cryptosystem as $\langle u \rangle$, the main property of the Paillier cryptosystem ensures that for arbitrary numbers u and v, we have $\langle u \rangle . \langle v \rangle = \langle u + v \rangle$. Therefore, $\langle h_l \rangle = \prod_{i \in I_L} \langle h_i \rangle$ and $\langle g_l \rangle = \prod_{i \in I_L} \langle g_i \rangle$. Consequently, the best split can be found in the following way. First, each passive party computes $\langle g_l \rangle$ and $\langle h_l \rangle$ for all possible splits locally, which are then sent back to the active party. The active party deciphers all $\langle g_l \rangle$ and $\langle h_l \rangle$ and calculates the global optimal split according to (3). We adopt the approximation scheme used by Chen and Guestrin,[30] so as to alleviate the need of enumerating all possible split candidates and communicating their $\langle g_i \rangle$ and $\langle h_i \rangle$. The details of our secure gradient aggregation algorithm are shown in Algorithm 1.

Algorithm 2. Split Finding

```
Input: I, instance space of current node
Input: $\{\mathbf{G}^i, \mathbf{H}^i\}_{i=1}^{m}$, aggregated encrypted gradient
       statistics from $m$ parties
Output: Partition current instance space according to the
        selected attribute's value
1:  /*Conduct on Active Party*/
2:  $g \leftarrow \sum_{i \in I} g_i, h \leftarrow \sum_{i \in I} h_i$
3:  for $i = 0$ to $m$ do
4:      for $k = 0$ to $d_i$ do
5:          $g_l \leftarrow 0, h_l \leftarrow 0$
6:          //enumerate all threshold value
7:          for $v = 0$ to $l_k$ do
8:              get decrypted values $D(\mathbf{G}_{kv}^i)$ and $D(\mathbf{H}_{kv}^i)$
9:              $g_l \leftarrow g_l + D(\mathbf{G}_{kv}^i), h_l \leftarrow h_l + D(\mathbf{H}_{kv}^i)$
10:             $g_r \leftarrow g - g_l, h_r \leftarrow h - h_l$
11:             $score \leftarrow \max(score, \frac{g_l^2}{h_l+\lambda} + \frac{g_r^2}{h_r+\lambda} - \frac{g^2}{h+\lambda})$
12:         end for
13:     end for
14: end for
15: Return $k_{opt}$ and $v_{opt}$ to the passive party $i_{opt}$ when we
    obtain the max score.
16: /*Conduct on Passive Party $i_{opt}$*/
17: Determine the selected attribute's value according to $k_{opt}$
    and $v_{opt}$ and partition current instance space.
18: Record the selected attribute's value and return [record
    id, $I_L$] back to the active party.
19: /*Conduct on Active Party*/
20: Split current node according to $I_L$ and associate current
    node with [party id, record id].
```

Following observation 1), the split-finding algorithm remains largely the same as XGBoost except for minor adjustments to fit the FL framework. Due to separation in features, SecureBoost requires different parties to store certain information for each split, so as to perform prediction for new samples. Passive parties should keep a lookup table, as shown in Figure 3. It contains split thresholds [feature id k, threshold value v] and a unique record id r used to index the table, in order to look up split conditions during inference. In the meantime, because the active party does not have features located in passive parties, for the active party to know which passive party to deliver an instance to, as well as instructing the passive party which split condition to use at inference time, it associates



Party 1 (Passive Party) — Training Set

Example	BIll Payment	Education
X1	3102	2
X2	17250	3
X3	14027	2
X4	6787	1
X5	280	1

Party 2 (Active Party) — Training Set

Example	Age	Gender	Marriage	Label
X1	20	1	0	0
X2	30	1	1	1
X3	35	0	1	1
X4	48	0	1	2
X5	10	1	0	3

Party 3 (Passive Party) — Training Set

Example	Amount of given credit
X1	5000
X2	300000
X3	250000
X4	300000
X5	200

Predict

Example	BIll Payment	Education
X6	4367	2

Example	Age	Gender	Marriage	Label
X6	28	1	0	0

Example	Amount of given credit
X6	5500

Lookup table

Party 1:

Record ID	Feature	threshold value
1	Bill Payment	5000

Party 2:

Record ID	Feature	threshold value
1	Age	40

Party 3:

Record ID	Feature	threshold value
1	Amount of given credit	800

FIGURE 3. An illustration of federated inference.

every tree node with a pair (party id i, record id r). Specific details about the split finding algorithm for SecureBoost are summarized in Algorithm 2. The problem remaining is the computation of optimal leaf weights. According to (4), the optimal weight of leaf j only depends on $\sum_{i \in I_j} g_i$ and $\sum_{i \in I_j} h_i$. Consequently, it follows similar procedures as split finding. When a leaf node is reached, the passive party sends $\langle \sum_{i \in I_j} g_i \rangle$ and $\langle \sum_{i \in I_j} h_i \rangle$ to the active party, which are then deciphered to compute corresponding weights using (4).

FEDERATED INFERENCE

In this section, we describe how the learned model (distributed among parties) can be used to classify a new instance even though the features of the instance to be classified are private and distributed among parties. Since each party knows its own features but nothing of the others, we need a secure distributed inference protocol to control passes from one party to another, based on the decision made. To illustrate the inference process, we consider a system with three parties, as depicted in Figure 3. Specifically, party 1 is the active party, which collects information including user's *monthly bill payment, education*, as well as the *label*, whether user X made the payment on time. Party 2 and party 3 are passive parties, holding features *age, gender, marriage status*, and *amount of given credit*, respectively. Suppose we wish to predict whether a user X_6 would make payment on time, then all sites would have to collaborate to make the prediction. The whole process is coordinated by the active party. Starting from the root, by referring to the record [party id:1, record id:1], the active party knows party 1 holds the root node, thereby requiring party 1 to retrieve the corresponding attribute, *Bill Payment*, from its lookup table based on record id 1. Since the classifying attribute is *bill payment* and party 1 knows that the bill payment for user $X6$ is 4,367, which is less than the threshold 5,000, it makes the decision that it should move down to its left child, node 1. Then, active party refers to the record [party id:3, record id:1] associated with node 1 and requires party 3 to conduct the same operations. This process continues until a leaf is reached.

THEORETICAL ANALYSIS FOR LOSSLESS PROPERTY

Theorem 1. SecureBoost is lossless, i.e., SecureBoost model M and XGBoost model M' would behave identically provided that models M and M' are identically initialized and hyperparameterized.

Proof. According to (3), g_l and h_l are the only information needed for the calculation of the best split, which can be obtained with encrypted g_i and h_i using the Paillier cryptosystem in SecureBoost. In the Paillier cryptosystem, the encryption of a message m is $\langle m \rangle = g^m r^n \bmod n^2$, for some random $r \in \{0, \ldots, n-1\}$. Given the definition of encrypted message, we have $\langle m_1 \rangle . \langle m_2 \rangle = \langle m_1 + m_2 \rangle$ for arbitrary message m_1 and m_2 under the Paillier cryptosystem, which can be proved as follows:

$$\begin{aligned}\langle m_1\rangle.\langle m_2\rangle &= (g^{m_1}r_1^c)(g^{m_2}r_2^c) \bmod n \\ &= g^{m_1+m_2}(r_1r_2)^c \bmod n = \langle m_1+m_2\rangle. \qquad (5)\end{aligned}$$

Therefore, we have $\langle h_l\rangle = \prod_{i\in I_L}\langle h_i\rangle$ and $\langle g_l\rangle = \prod_{i\in I_L}\langle g_i\rangle$. Provided that with the same initialization, an instance i will have the same value of g_i and h_i under either setting. Thus, models M and M' can always achieve the same best split throughout the construction of the tree and result in identical M and M', which ensures the property of lossless. □

SECURITY DISCUSSION

SecureBoost avoids revealing data records held by each of the parties to others during training and inference, thus protecting the privacy of individual parties' data. However, we stress that there is some leakage that can be inferred during the protocol execution, which is quite different for passive versus active parties.

The **active party** is in an advantageous position with SecureBoost as it learns the instance space for each split and which party is responsible for the decision at each node. Also, it learns all the possible values of g_l, g_r and h_l, h_r, during learning. The former seems unavoidable in this setting, unless one is willing to severely increase the overhead during the inference phase. However, the latter can be avoided using secure multiparty computation techniques for comparison of encrypted values (e.g., see the works of Bost *et al.*[32] and Baldimtsi *et al.*[33]). In this way, the active party learns only the optimal g_l, g_r, h_l, h_r per party; on the other hand, this significantly affects the efficiency during learning.

Note that the instances that are associated with the same leaf strongly indicates that they belong to the same class. We denote the proportion of samples that belong to the majority class as **leaf purity**. The information leakage with respect to **passive parties** is directly related to **leaf purity** of the first tree of SecureBoost. Moreover, the first tree's **leaf purity** can be inferred from the weight of the leaves.

Theorem 2. Given a learned SecureBoost model, its first tree's leaf purity can be inferred from the weight of the leaves.

Proof. The loss function for binary classification problem is given as follows:

$$L = y_i \log(1+e^{-\hat{y}_i}) + (1-y_i)\log(1+e^{\hat{y}_i}). \qquad (6)$$

Based on the loss function, we have $g_i = \hat{y}_i^{(0)} - y_i$ and $h_i = \hat{y}_i^{(0)} * (1-\hat{y}_i^{(0)})$ during the construction of the decision tree at first iteration. Specifically, $\hat{y}_i^{(0)}$ is given as an initialized value. Suppose we initialize all $\hat{y}_i^{(0)}$ as a where $0 < a < 1$.

According to (4), for the instances associated with the specific leaf j, $\hat{y}_i^{(1)} = S(w_j^*) = S(-\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i+\lambda})$ where $S(x)$ is the sigmoid function. Suppose the number of instances associated with leaf j is n_j and the percentage of positive samples is θ_j. When n_j is relatively big, we can ignore λ in $-\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i+\lambda}$ and rewrite the weight of leaf j as $w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i} = -\frac{\theta_j*n*(a-1)+(1-\theta_j)*n*a}{n*a*(1-a)} = -\frac{\theta_j*n*(a-1)+(1-\theta_j)*n*a}{n*a*(1-a)} = \frac{a-\theta_j}{a(a-1)}$. By reformulating the equation, we have $\theta_j = a - a(a-1)w_j^*$. θ_j depends on a and w_j^* and a is given as initialization. Thus, w_j^* is the key to determine θ_j. Note that θ_j can be used to represent the leaf purify of leaf j (i.e., purify of leaf j can be formally written as $\max(\theta_j, 1-\theta_j)$, and the leaf purity of the first tree can be inferred from the weight of the leaves (w_j^*), given a learned SecureBoost model. □

According to Theorem 2, given a SecureBoost model, the weight of the leaves of its first tree can reveal sensitive information. In order to reduce information leakage with respect to **passive parties**, we opt to store decision tree leaves at the active party and propose a modified version of our framework, called *Reduced-Leakage SecureBoost (RL-SecureBoost)*. With *RL-SecureBoost*, the active party learns the first tree independently based only on its own features, which fully protects the instance space of its leaves. Hence, all the information that passive parties learn is based on residuals. Although the residuals may also reveal information, we prove that as the purity in the first tree increases, this residual information decreases.

Theorem 3. As the purity in the first tree increases, the residual information decreased.

Proof. As mentioned before, for binary classification problem, we have $g_i = \hat{y}_i^{(t-1)} - y_i$ and $h_i = \hat{y}_i^{(t-1)} * (1-\hat{y}_i^{(t-1)})$, where $g_i \in [-1, 1]$. Hence,

$$\begin{cases} h_i = g_i(1-g_i), & \text{if } y_i = 0 \\ h_i = -g_i(g_i+1), & \text{if } y_i = 1. \end{cases} \qquad (7)$$

When we construct the decision tree at the tth iteration with k leaves to fit the residuals of the previous tree, in essential, we split the data into k clusters to minimize the following loss:

$$\begin{aligned} L &= -\sum_{j=1}^{k} \frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i} \\ &= -\sum_{j=1}^{k} \frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j^N} g_i(1-g_i) + \sum_{i\in I_j^P} -g_i(1+g_i)}. \end{aligned} \tag{8}$$

We know $\hat{y}_i^{(t-1)} \in [0,1]$ and $g_i = \hat{y}_i^{(t-1)} - y_i$. Thus, we have $g_i \in [-1,0]$ for positive samples and $g_i \in [0,1]$ for negative samples. Taking the range of g_i into consideration, we can rewrite (8) as follows:

$$\sum_{j=1}^{k} \frac{\left(\sum_{i\in I_j^N} |g_i| - \sum_{i\in I_j^P} |g_i|\right)^2}{\sum_{i\in I_j^N} |g_i|(|g_i|-1) + \sum_{i\in I_j^P} |g_i|(|g_i|-1)} \tag{9}$$

where I_j^N and I_j^P denote the set of negative samples and positive samples associated with leaf j, respectively. We denote the expectation of $|g_i|$ for positive samples as μ_p and the expectation of $|g_i|$ for negative samples as μ_n. When we have a large amount of samples but small number of leave nodes k, we can use the following equation to approximate (9):

$$\sum_{j=1}^{k} \frac{(n_j^n \mu_n - n_j^p \mu_p)^2}{n_j^n \mu_n(\mu_n - 1) + n_j^p \mu_p(\mu_p - 1)} \tag{10}$$

where n_j^n and n_j^p represent the number of negative samples and positive samples associated with leaf j, respectively. Since $\mu_n \in [0,1]$ and $\mu_n \in [0,1]$, we know the numerator has to be positive and the denominator has to be negative. Thus, the whole equation has to be negative. Minimizing (10) is equal to maximizing the numerator while minimizing the denominator. Note that the denominator is $\sum x^2$ and the numerator is $(\sum x)^2$, where $x \in [0,1]$. The equation is dominated by numerator. Thereby, minimizing (10) can be regarded as maximizing the numerator $(n_j^n \mu_n - n_j^p \mu_p)^2$. Ideally, we require $n_j^n = n_j^p$ in order to prevent label information from divulging. When $|\mu_n - \mu_p|$ is bigger, more possible we can achieve the goal. And we know $|g_i| = |\hat{y}_i^{(t-1)} - y_i| = \hat{y}_i^{(t-1)}$ for negative samples and $|g_i| = |\hat{y}_i^{(t-1)} - y_i| = 1 - \hat{y}_i^{(t-1)}$ for positive samples. Thereby, $\mu_n = \frac{1}{N_n}\sum_{j=1}^{k}(1-\theta_j)n_j\hat{y}_i^{(t-1)}$ and $\mu_p = \frac{1}{N_p}\sum_{j=1}^{k}\theta_j n_j(1-\hat{y}_i^{(t-1)})$. $|\mu_n - \mu_p|$ can be calculated as follows:

$$\begin{aligned} |\mu_n - \mu_p| = |&\frac{1}{N_n}\sum_{j=1}^{k}(1-\theta_j)n_j\hat{y}_i^{(t-1)} \\ &- \frac{1}{N_p}\sum_{j=1}^{k}\theta_j n_j(1-\hat{y}_i^{(t-1)})| \end{aligned} \tag{11}$$

where N_n and N_p, respectively, correspond to the number of negative samples and positive samples in total, θ_j is the percentage of positive samples associated with leaf j for decision tree at $(t-1)$th iteration (previous decision tree), n_j denotes the number of instances associated with leaf j for previous decision tree, and $\hat{y}_i^{(t-1)} = S(w_j)$, where w_j represents the weight of the jth leaf of previous decision tree. When the positive samples and negative samples are balanced, $N_n = N_p$, we have

$$\begin{aligned} &|\mu_n - \mu_p| \\ &\quad = \frac{1}{N_n}|\sum_{j=1}^{k}((1-\theta_j)n_j S(w_j) - \theta_j n_j(1-S(w_j))| \\ &\quad = \frac{1}{N_n}\sum_{j=1}^{k} n_j|(S(w_j) - \theta_j)| \\ &\quad = \frac{1}{N_n}\sum_{j=1}^{k} n_j|(S(\frac{a-\theta_j}{a(a-1)}) - \theta_j)|. \end{aligned} \tag{12}$$

As observed from (12), it achieves the minimum value when $S(\frac{a-\theta_j}{a(a-1)}) = a$. By solving the equation, we have the optimal solution of θ_j as $\theta_j* = a(1+(1-a)\ln(\frac{a}{1-a}))$. In order to achieve bigger $\mu_n - \mu_p$, we want the deviations from θ_j to θ_j* to be as big as possible. When we have proper initialization of a, for instance $a = 0.5$, $\theta_j* = 0.5$. In this case, maximizing $|\theta_j - \theta_j*|$ is the same as maximizing $\max(\theta_j, 1-\theta_j)$, which exactly is the leaf purity. Therefore, we have proved that high leaf purity will guarantee big difference between μ_n and μ_p, which finally results in less information leakage. We complete our proof. □

Given Theorem 3, we prove that *RL-SecureBoost* is secure as long as its first tree learns enough information to mask the actual label with residuals. Moreover, as we experimentally demonstrate in the "Experiments" section, *RL-SecureBoost* performs identically as *SecureBoost* in terms of prediction accuracy.

EXPERIMENTS

We conduct experiments on two public datasets.

Credit 1[b]: It involves the problem of classifying whether a user would suffer from serious financial problems. It contains a total of 150,000 instances and 10 attributes.

[b]https://www.kaggle.com/c/GiveMeSomeCredit/data

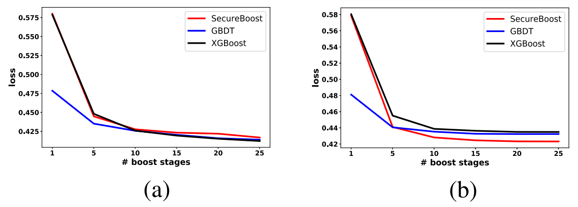


FIGURE 4. Loss convergence. (a) Learning Curve. (b) Test Error.

Credit 2[c]: It is also a credit scoring dataset, correlated to the task of predicting whether a user would make payment on time. It consist of 30,000 instances and 25 attributes in all.

In our experiment, we use $2/3$ of each dataset for training and the remaining for testing. We split the data vertically into two halves and distribute them to two parties. To fairly compare different methods, we set the maximum depth of each tree as 3, the fraction of samples used to fit individual regression trees as 0.8, and learning rate as 0.3 for all methods. The Paillier encryption scheme is taken as our encryption scheme with a key size of 512 bits. All experiments are conducted on a machine with 8 GB RAM and Intel Core i5-7200u CPU.

Scalability

Note that the efficiency of SecureBoost may be reflected by the rate of convergence and runtime, which may be influenced by 1) the maximum depth of individual regression trees and 2) the size of the datasets. In this section, we conduct convergence analysis as well as study the impact of all the variables on the runtime of learning. All experiments are conducted on dataset Credit 2.

First, we are interested in the convergence rate of our proposed system. We compare the convergence behavior of SecureBoost with nonfederated tree-boosting counterparts, including GBDT[d] and XGBoost.[e] As can be observed from Figure 4, SecureBoost shows a similar learning curve to other nonfederated methods on the training dataset and even performs slightly better than others on the test dataset. In addition, the convergence behavior of training and test losses of SecureBoost are very much alike GBDT and XGBoost.

[c]https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset
[d]http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
[e]https://github.com/dmlc/xgboost

TABLE 1. First tree versus second tree in terms of leaf purity.

Mean Purity	Credit 1	Credit 2
1st Tree	0.8058	0.7159
2rd Tree	0.66663	0.638

Next, to investigate how the maximum depth of individual trees affects the runtime of learning, we vary the maximum depth of an individual tree among $\{3, 4, 5, 6, 7, 8\}$ and report the runtime of one boosting stage. As depicted in Figure 5(a), the runtime increases almost linearly with the maximum depth of individual trees, which indicates that we can train deep trees with relatively little additional time, which is very appealing in practice, especially in scenarios like big data.

Finally, we study the impact of data size on the runtime of our proposed system. We augment the feature sets by feature products. We fix the maximum depth of individual regression trees to 3 and vary the feature number in $\{50, 500, 1000, 5000\}$ and the sample number in $\{5000, 10000, 30000\}$. We compare the runtime of one boosting stage to investigate how each variant affects the efficiency of the algorithm. We make similar observations on both Figure 5(b) and (c), which imply that sample and feature numbers contribute equally to running time. In addition, we can see that our proposed framework scales well even with relatively big data.

Performance of RL-SecureBoost

To investigate the performance of RL-SecureBoost in both security and prediction accuracy, we aim to answer the following questions: 1) Does the first tree, built upon only features held by the active party, learns enough information to reduce information leakage? 2) Does RL-SecureBoost suffer from a loss of accuracy compared with SecureBoost?

First, we study the performance of RL-SecureBoost in security. Following the analysis in the "Security Discussion" section, we evaluate information leakage in terms of leaf purity. Also, we know that as the leaf purity in the first tree increases, leaked information is reduced. Thereby, to verify the security of RL-SecureBoost, we have to illustrate that the first tree of RL-SecureBoost performs well enough to reduce the information leaked from the second tree. As shown in Table 1, we compare the mean leaf purity of the first tree with that of the second tree. In particular, the mean leaf purity is the weighted average, which is calculated by $\sum_{i=0}^{k} \frac{n_i}{n} p_i$. Here, k and n, respectively, represent number of leaves and number of instances in

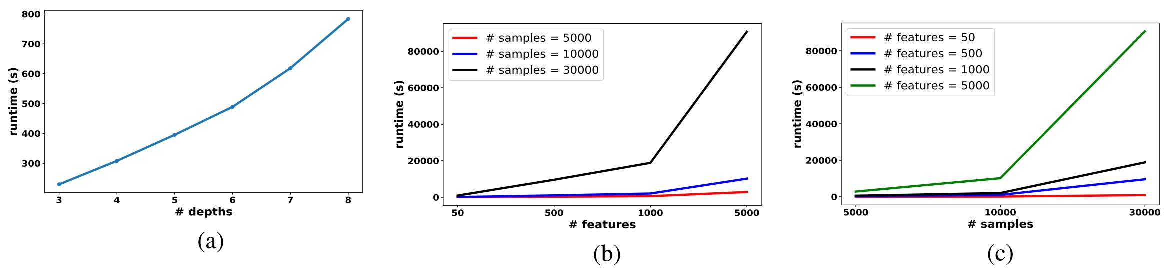


FIGURE 5. Scalability Analysis of SecureBoost. (a) Runtime w.r.t. maximum depth of individual tree. (b) Runtime w.r.t. feature size. (c) Runtime w.r.t. sample size.

TABLE 2. Classification performance for RL-SecureBoost versus SecureBoost.

Model	Credit 1			Credit 2		
	ACC	F1-score	AUC	ACC	F1-score	AUC
1st Tree, SecureBoost	0.9298	0.012	0.7002	0.7806	0	0.6381
1st Tree, RL-SecureBoost	0.9186	0	0.6912	0.7793	0	0.6320
Overall, SecureBoost	0.9345	0.2576	0.8461	0.8180	0.4634	0.7701
Overall, RL-SecureBoost	0.9331	0.2549	0.8423	0.8179	0.4650	0.7682

total. p_i and n_i are, respectively, defined as leaf purity and number of instances associated with leaf i. According to Table 1, the mean leaf purity decreases significantly from the first to the second tree on both datasets, which reflects a great reduction in information leakage. Moreover, the mean leaf purity of the second tree is just over 0.6 on both datasets, which is good enough to ensure a safe protocol.

Next, to investigate the prediction performance of RL-SecureBoost, we compare RL-SecureBoost with SecureBoost with respect to the first tree's performance and the overall performance. We consider commonly used metrics including accuracy, Area under ROC curve (AUC), and f1-score. The results are presented in Table 2. As observed, RL-SecureBoost performs equally well compared to SecureBoost in almost all cases. We also conduct a pairwise Wilcoxon signed-rank test between RL-SecureBoost and SecureBoost. The comparison results indicate that RL-SecureBoost is as accurate as SecureBoost, with a significance level of 0.05. The property of lossless is still guaranteed for RL-SecureBoost.

CONCLUSION

In this article, we proposed a lossless privacy-preserving tree-boosting algorithm, SecureBoost, to train a high-quality tree-boosting model with private data split across multiple parties. We theoretically prove that our proposed framework is as accurate as non-federated gradient tree-boosting counterparts. In addition, we analyze information leakage during the protocol execution and propose provable ways to reduce it.

ACKNOWLEDGMENTS

This work was partially supported by the National Key Research and Development Program of China under Grant 2018AAA0101100.

REFERENCES

1. IT GOVERNANCE PRIVACY TEAM. *EU General Data Protection Regulation (GDPR), 3rd ed.: An Implementation and Compliance Guide*. Ely, Cambridgeshire, United Kingdom: IT Governance Publishing, 2019. Accessed: Jun. 4, 2021, doi: 10.2307/j.ctvr7fcwb.
2. J. P. Albrecht, "How the GDPR will change the world," *Eur. Data Protection Law Rev.*, vol. 2, pp. 287–289, 2016, doi: 10.21552/EDPL/2016/3/4.
3. V. Mayer-Schonberger and Y. Padova, "Regime change: Enabling big data through Europe's new data protection regulation," *Columbia Sci. Technol. Law Rev.*, vol. 17, pp. 315–335, 2015.

4. B. Goodman and S. Flaxman, “European Union regulations on algorithmic decision-making and a ‘right to explanation’,” 2016, *arXiv:1606.08813*, doi: 10.1609/aimag.v38i3.2741.
5. S. Hardy *et al.*, “Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption,” 2017, *arXiv:1711.10677*.
6. P. Mohassel and Y. Zhang, “Secureml: A system for scalable privacy-preserving machine learning,” in *Proc. 38th IEEE Symp. Secur. Privacy*, 2017, pp. 19–38, doi: 10.1109/SP.2017.12.
7. C. Dwork *et al.* “The algorithmic foundations of differential privacy,” *Foundations Trends Theor. Comput. Sci.*, vol. 9, no. 3-4, pp. 211–407, 2014, doi: 10.1561/0400000042.
8. C. Dwork, “Differential privacy: A survey of results,” in *Proc. Int. Conf. Theory Appl. Models Comput.*, 2008, pp. 1–19, doi: 10.1007/978-3-540-79228-4_1.
9. J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” 2016, *arXiv:1610.05492*.
10. K. Bonawitz *et al.*, “Practical secure aggregation for privacy-preserving machine learning,” in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1175–1191, doi: 10.1145/3133956.3133982.
11. Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning: Concept and applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 12:1–12:19, 2019, doi: 10.1145/3298981.
12. Q. Yang, Y. Liu, Y. Cheng, T. Chen, and H. Yu, *Federated Learning*. San Rafael, CA, USA: Morgan & Claypool, 2019.
13. Y. Liu *et al.*, “A communication efficient vertical federated learning framework,” 2019. [Online]. Available: http://arxiv.org/abs/1912.11187
14. X. Liang, Y. Liu, J. Luo, Y. He, T. Chen, and Q. Yang, “Self-supervised cross-silo federated neural architecture search,” 2021, *arXiv:2101.11896*.
15. J. Vaidya, “A survey of privacy-preserving methods across vertically partitioned data,” in *Privacy-Preserving Data Mining*. Berlin, Germany: Springer, 2008, pp. 337–358, doi: 10.1007/978-0-387-70992-5_14.
16. J. Vaidya and C. Clifton, “Privacy-preserving decision trees over vertically partitioned data,” in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2005, pp. 139–152, doi: 10.1007/11535706_11.
17. R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, 2015, pp. 1310–1321, doi: 10.1145/2810103.2813687.
18. P. Kairouz *et al.*, “Advances and open problems in federated learning,” 2019, [Online]. Available: http://arxiv.org/abs/1912.04977, doi: 10.1561/2200000083.
19. L. Zhao *et al.*, “Inprivate digging: Enabling tree-based distributed data mining with differential privacy,” in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 2087–2095, doi: 10.1109/INFOCOM.2018.8486352.
20. Q. Li, Z. Wen, and B. He, “Practical federated gradient boosting decision trees,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, 2020, pp. 4642–4649, doi: 10.1609/AAAI.V34I04.5895.
21. T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” 2018, *arXiv:1812.06127*.
22. F. Hanzely, S. Hanzely, S. Horváth, and P. Richtárik, “Lower bounds and optimal algorithms for personalized federated learning,” 2020, *arXiv:2010.02372*.
23. M. Mohri, G. Sivek, and A. T. Suresh, “Agnostic federated learning,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4615–4625.
24. M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, “Bayesian nonparametric federated learning of neural networks,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7252–7261.
25. H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” 2020, *arXiv:2002.06440*.
26. Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, “A secure federated transfer learning framework,” *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 70–82, Jul./Aug. 2020, doi: 10.1109/MIS.2020.2988525.
27. J. Vaidya, C. Clifton, M. Kantarcioglu, and A. S. Patterson, “Privacy-preserving decision trees over vertically partitioned data,” *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, 2008, Art. no. 14, doi: 10.1007/11535706_11.
28. M. Djatmiko *et al.*, “Privacy-preserving entity resolution and logistic regression on encrypted data,” *Private Secure Mach. Learn.*, 2017.
29. G. Liang and S. S. Chawathe, “Privacy-preserving inter-database operations,” in *Proc. Int. Conf. Intell. Secur. Informat.*, 2004, pp. 66–82, doi: 10.1007/978-3-540-25952-7_6.
30. T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.
31. P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238, doi: 10.1007/3-540-48910-X_16.

32. R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *Proc. 22nd Annu. Netw. Distrib. Syst. Secur. Symp.*, 2015, pp. 331–345, doi: 10.14722/NDSS.2015.23241.
33. F. Baldimtsi, D. Papadopoulos, S. Papadopoulos, A. Scafuro, and N. Triandopoulos, "Server-aided secure computation with off-line parties," in *Proc. Comput. Secur. 22nd Eur. Symp. Res. Comput. Secur., Proc., Part I*, 2017, pp. 103–123, doi: 10.1007/978-3-319-66402-6_8.

KEWEI CHENG is a Ph.D. student at the University of California, Los Angeles, Los Angeles, CA, USA. Her research interests include knowledge graph reasoning, machine learning, and federated learning. She received the B.Eng. degree in software engineering from Sichuan University, Chengdu, China and the M.Sc. degree in computer science from Arizona State University, Tempe, AZ, USA. She is the corresponding author of this article. Contact her at keweicheng@g.ucla.edu.

TAO FAN is a Principal Researcher with the AI Department, WeBank, ShenZhen, China. He is now responsible for FATE, an industrial-level federated learning open-source project. He received the master's degree from the University of Science and Technology of China, Hefei, China. Contact him at dylanfan@webank.com.

YILUN JIN is a Ph.D. student at the Hong Kong University of Science and Technology, Hong Kong. His research focuses on machine learning and data mining, including federated learning, transfer learning, and learning on graph data. He received bachelor's degrees in computer science and economics from Peking University, Beijing, China. Contact him at yilun.jin@connect.ust.hk.

YANG LIU is a Principal Researcher with the AI Department, WeBank, Shenzhen, China. She holds multiple patents. Her research interests include federated learning, transfer learning, multiagent systems, statistical mechanics, and applications of these technologies in the industry. She received the Ph.D. degree from Princeton University, Princeton, NJ, USA. Contact her at liuyang.princeton@gmail.com.

TIANJIAN CHEN is the Deputy General Manager with the AI Department, WeBank, Shenzhen, China. He is now responsible for building the Banking Intelligence Ecosystem based on Federated Learning Technology. Contact him at tobychen@webank.com.

DIMITRIOS PAPADOPOULOS is an Assistant Professor with the Computer Science and Engineering Department, Hong Kong University of Science and Technology, Hong Kong. His research focuses on the development of novel cryptographic protocols for a variety of application scenarios and problems related to cloud data privacy, distributed machine learning, and blockchain settings. He received the Ph.D. degree in computer science from Boston University, Boston, MA, USA. Contact him at dipapado@cse.ust.hk.

QIANG YANG is the Head of AI at WeBank, Shenzhen, China, Chief AI Officer, and Chair Professor with the Computer Science and Engineering Department, Hong Kong University of Science and Technology, Hong Kong. His research interests include transfer learning, automated planning, federated learning, and case-based reasoning. He received the Ph.D. degree from the Computer Science Department, University of Maryland, College Park, MD, USA. He is a Fellow of ACM, AAAI, IEEE, IAPR, and AAAS. Contact him at qyang@cse.ust.hk.