SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

August 2024, Vol. 67, Iss. 8, 182303:1–182303:16
https://doi.org/10.1007/s11432-023-4013-x

Multi-party privacy-preserving decision tree training with a privileged party

Yiwen TONG[1], Qi FENG[1*], Min LUO[1,2] & Debiao HE[1,3*]

[1]*Key Laboratory of Aerospace Information Security and Trusted Computing Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan* 430072*, China;*
[2]*Shanghai Technology Innovation Centre of Distributed Privacy-Preserving Artificial Intelligence, Matrix Elements Technologies, Shanghai* 200232*, China;*
[3]*Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center, Qilu University of Technology (Shandong Academy of Sciences), Jinan* 250014*, China*

Received 2 December 2023/Revised 4 March 2024/Accepted 22 April 2024/Published online 23 July 2024

Abstract Currently, a decision tree is the most commonly used data mining algorithm for classification tasks. While a significant number of studies have investigated privacy-preserving decision trees, the methods proposed in these studies often have shortcomings in terms of data privacy breach or efficiency. Additionally, these methods typically only apply to symmetric frameworks, which consist of two or more parties with equal privilege, and are not suitable for asymmetric scenarios where parties have unequal privilege. In this paper, we propose SecureCART, a three-party privacy-preserving decision tree training scheme with a privileged party. We adopt the existing pMPL framework and design novel secure interactive protocols for division, comparison, and asymmetric multiplication. Compared to similar schemes, our division protocol is 93.5–560.4× faster, with the communication overhead reduced by over 90%; further, our multiplication protocol is approximately 1.5× faster, with the communication overhead reduced by around 20%. Our comparison protocol based on function secret sharing maintains good performance when adapted to pMPL. Based on the proposed secure protocols, we implement SecureCART in C++ and analyze its performance using three real-world datasets in both LAN and WAN environments. he experimental results indicate that SecureCART is significantly faster than similar schemes proposed in past studies, and that the loss of accuracy while using SecureCART remains within an acceptable range.

Keywords privacy protection, decision trees, secure multi-party computation, secret sharing, privileged party

1 Introduction

It is observed that a significant number of studies have explored privacy-preserving data mining (PPDM). Decision trees, as one of the most popular data mining algorithms, have gained widespread acclaim for their ability to process a wide range of tasks in machine learning and data analysis. The training of decision trees, similar to that of other data mining algorithms, requires a large number of datasets that are often owned by different organizations, such as government agencies, corporations, and so on, that may have different levels of hierarchy and authority. Due to regulatory requirements as well as associated individuals' inclinations, these organizations do not allow the use of their confidential raw data for interactions with other parties. Hence, they need to adopt secure multi-party learning (MPL) protocols for privacy-preserving decision tree training.

The majority of currently used MPL frameworks [1–4] are designed considering multiple parties having the same privileges. These frameworks are generally efficient, robust, and can tolerate the appearance of malicious parties or the withdrawal of parties. However, they cannot make distinctions regarding participant privileges. Therefore, in this paper, we propose a privacy-preserving scheme for decision tree training based on the recently proposed pMPL framework [5], that can be used in scenarios involving a privileged party.

[*] Corresponding author (email: fengqi.whu@whu.edu.cn, hedebiao@163.com)

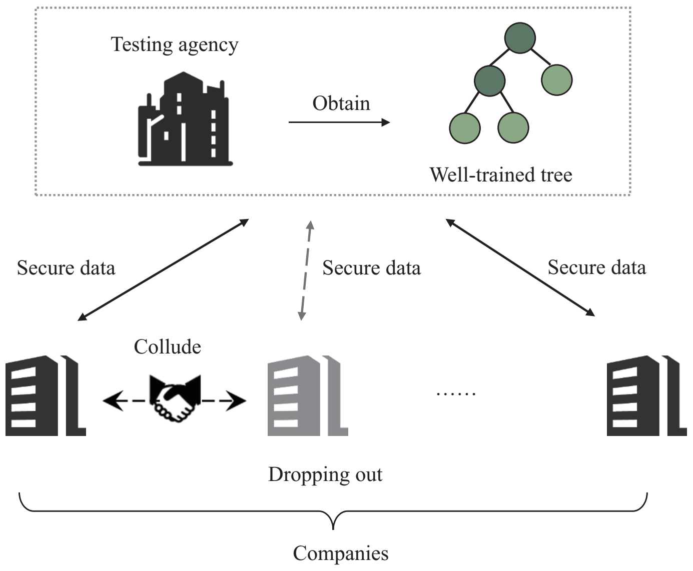

Figure 1 (Color online) Example scenario.

In the pMPL framework, parties are divided as follows: the privileged party and assistant parties, each with different functionalities. Consider the scenario in Figure 1 as an example. The testing agency is a government department with a supervisory function, and the companies work as data providers for the testing agency. The testing agency aims to complete decision tree training based on its own dataset and the dataset owned by the companies. According to the pMPL framework, the testing agency is regarded as a privileged party, that usually acts as the initiator of the machine learning task, with the privilege of obtaining the final training results, while the companies are regarded as assistant parties; during training, these companies are allowed to drop out or collude with each other. Parties secret-share their raw data, and interact with each other using secure shares.

Although pMPL can fulfill the requirements of our scenario, its building blocks are insufficient to complete privacy-preserving decision tree training. For example, the division operation for calculating the Gini impurity and the comparison operation for comparing the Gini impurity in the decision tree training algorithm are not provided in [5]. Therefore, in this study, we first focus on the design of new building blocks that ensure privacy and efficiency during decision tree training. The main challenge during the design of these blocks is that division protocols aiming for high precision often involve complex processes and have lower efficiency, whereas more efficient division protocols may sacrifice precision. Hence, balancing the extent of efficiency and precision is a critical consideration. Moreover, the recently proposed comparison protocols [6, 7] based on function secret sharing (FSS) have demonstrated good performance, but designing an FSS-based comparison protocol adapted to the unique framework of pMPL is a challenging task. However, the design of new building blocks overcomes these challenges, and we ensure both efficiency and data security when implementing a pMPL-based privacy-preserving decision tree training method.

1.1 Our contributions

In this paper, we develop new building blocks considering the pMPL framework. Subsequently, we design a privacy-preserving decision tree training scheme based on these blocks. The key contributions of this paper are listed below.

• We design several novel protocols for basic operations for the pMPL framework, including secure division, secure comparison and secure asymmetric multiplication; these protocols show significant performance improvement as compared to the existing protocols.

• We propose SecureCART, a privacy-preserving decision tree training scheme with a privileged party by combining the designed building blocks of pMPL, that ensures security against semi-honest adversaries while maintaining efficiency and robustness.

• We perform experiments on various datasets and compare SecureCART with similar schemes proposed in past studies. The results provide strong evidence of the efficiency of SecureCART.

1.2 Paper organization

The remainder of this paper is structured as follows. Section 2 discusses the relevant works. Section 3 provides a concise overview to the preliminaries. Section 4 outlines the system model and threat model, and also explains our design goals. Sections 5 and 6 include the design details of building blocks and their implementation to realize a multi-party privacy-preserving decision tree training scheme with a privileged party. The complete implementation and performance evaluation results of the proposed protocols and model are given in Section 7. Finally, Section 8 presents the conclusion and provides insights for further research.

2 Related work

Studies on privacy protection for decision trees typically focus on processes such as training and evaluation, in which training is usually achieved by classical algorithms including ID3 [8], C4.5 [9] and classification and regression tress (CART) [10].

In a multi-party environment, the entire dataset can be partitioned in three distinct ways: horizontal partitioning, vertical partitioning, and arbitrary partitioning.

In horizontal partitioning, each party holds the same features but different subsets of instances. Lindell et al. [11] pioneered a two-party privacy-preserving decision tree algorithm for horizontally partitioned data, that achieved perfect security through cryptography-based techniques such as oblivious transfer; however, this algorithm had significant computational overhead. Subsequently, several other improved solutions [12–15] for horizontally partitioned data were proposed. In arbitrary partitioning, the dataset can be partitioned both horizontally and vertically; and typical privacy-preserving decision tree schemes for arbitrary partitioning including [16–18].

In vertical partitioning, the dataset is divided into multiple subsets based on features, with each party owning distinct feature subsets. Since this study deals with vertically partitioned data, we will mainly focus on studies that have used vertically partitioned data. To the best of our knowledge, existing solutions have certain limitations that do not fully meet the requirements of our scenario.

Some of these studies have shortcomings in terms of privacy protection. Du et al. [19] introduced the first solution for a two-party privacy-preserving decision tree with vertical datasets, that privately calculated the information gain for ID3 decision tree training by masking the sensitive attributes with random numbers. However, the major drawbacks of this approach is that labels are disclosed to all parties. The methods proposed by She et al. [20] and Wang et al. [21] both focus on constructing decision trees on the joins of multiple private tables. While they achieve scalability, they do not ensure data privacy. Vaidya et al. proposed schemes [22, 23] for privacy-preserving ID3 algorithm with vertically partitioned datasets that offered better security; however, these schemes did not ensure complete data privacy as a risk of revealing intermediate results was attached to the scheme design. Additionally, Dansana et al. [24] designed a multi-party CART algorithm based on the secure sum protocol and intersection protocol, that attempted to minimize the information leakage. Hu et al. [25] also aimed to reduce data leakage, wherein users continued to have the opportunity to learn intermediate data such as split statistics. Cheng et al. [26] constructed SecureBoost using the Paillier homomorphic encryption algorithm. Although this approach avoided leakage of labels and features, it still shared the optimal feature splits with the label holder.

However, some of these studies have drawbacks in terms of efficiency. The scheme introduced in [22] faces certain performance bottlenecks regarding computing information gain. Recently, Abspole et al. [27] have presented an approach for achieving the privacy protection during decision trees training, that can handle all types of attributes. Although it adopted techniques such as secret sharing and a sorting network to efficiently compute the Gini impurity, the computational overhead remains high. Pivot, introduced by Wu et al. [28], mainly utilized homomorphic encryption to design a secure CART algorithm for semi-honest parties. This approach ensured zero leakage of private data; however, it also incurred high computational overhead due to the use of cryptographic techniques. Zheng et al. [29] designed Privet, a framework supporting privacy-preserving VFL service for gradient boosted decision tables. While this solution boasts of high accuracy, it requires improvement in terms of efficiency.

Additionally, some schemes have limitations apart from privacy and efficiency concerns. Chen et al. [6] designed PriVDT, which is an efficient two-party framework for vertical CART decision trees. This

Table 1 Notations

Notation	Description	Notation	Description
$[x]$ / $\langle x\rangle$	The shares of additive / vector space secret sharing	$F_j^{(i)}$	The dataset associated with the j-th feature held by P_i
x / $\boldsymbol{x}$	Scalar / vector	$m^{(i)}$	The number of features held by P_i
$\boldsymbol{x}\cdot\boldsymbol{y}$ / $\boldsymbol{x}\circ\boldsymbol{y}$	Dot product / element-wise product of vectors	$\delta_j^{(i)}$	The number of splits of the j-th feature held by P_i
$\alpha_0,\ldots,\alpha_3''$	The public constants used for reconstruction	$\boldsymbol{\gamma}$	The vector of current available samples
$\mathcal{P}$	The set of n parties	t	The number of training samples
$\varLambda$	The minimal access structure	$\boldsymbol{L}_\mu$	The vector of samples belonging to the μ-th class
A	The authorized subset of $\varLambda$	$\boldsymbol{C}_\mu$	The vector of available samples belonging to the μ-th class
$\mathcal{F}$	The set of features held by all parties	M	The number of classes

framework improved both data privacy and algorithm efficiency as compared to that of previous solutions; however, it required the involvement of a trusted third party. Therefore, considering the abovementioned disadvantages of the existing schemes, our proposed approach, that overcomes these disadvantages, has been discussed in terms of the designed improvements in Subsection 1.1.

3 Preliminaries

In this section, we explain the notations mentioned frequently throughout this paper, as shown in Table 1. Subsequently, we review the relevant decision tree algorithms briefly, with a particular focus on the training processes of decision trees. Finally, we introduce some of the underlying cryptographic techniques such as secret sharing.

3.1 Decision tree

The decision tree method stands out as a powerful machine learning algorithm adept at handling both classification and regression tasks. In this work, we employ the CART algorithm [10] to implement the classification tree training task. The CART decision tree is constructed recursively as a binary decision tree, and its features are selected with the Gini impurity minimization criteria. The CART algorithm is specifically demonstrated as follows.

(1) Assuming that the dataset of the available sample corresponding to the current node is D, the Gini impurity is calculated for all splits of the features. More specifically, for each possible split value sv of each feature, the current dataset D is divided into two subsets, D_l and D_r, by determining whether the feature value f_j of each sample equals to (or belongs to) sv; subsequently, the variant form of the Gini impurity is calculated by (1) as shown below:

$$\tilde{G}(D, f_j = \mathrm{sv}) = \frac{\sum_{\mu l=1}^{M}|C_{\mu l}^2|}{|D_l|} + \frac{\sum_{\mu r=1}^{M}|C_{\mu r}^2|}{|D_r|}, \tag{1}$$

where $|D_l|$ represents the count of samples for subset D_l, and $|C_{\mu l}|$ represents the count of samples that belongs to the class $C_{\mu l}$ in D_l, as well as $|D_r|$ and $|C_{\mu r}|$.

(2) The optimal split is selected based on the maximum value of $\tilde{G}$. With the optimal split and its corresponding feature, we can separate the samples into two parts, namely D_l and D_r, to generate a pair of child nodes of the current node. The left node's dataset is D_l, and the right node's dataset is D_r. After generating two child nodes, the value of the optimal split is discarded from consideration.

(3) Steps (1) and (2) are called recursively to the left and right child nodes until the stopping conditions are met (i.e., when the number of samples is below a certain threshold or there are no available features for further splitting).

Based on the training dataset, starting from the root node, we can eventually construct a well-trained CART decision tree by implementing the above steps.

3.2 Cryptographic primitives

3.2.1 *Vector space secret sharing*

We adhere to the definition of vector space secret sharing from [30]. The characteristic of vector space secret sharing lies in allowing parties of certain combinations to jointly reveal the secret value, while parties of other combinations cannot reconstruct the secret value despite colluding with each other.

Formally, let $\mathcal{P} = \{P_0, P_1, \ldots, P_{n-1}\}$ be a set of n parties, and $\varLambda$ be the minimal access structure, that contains the minimum sets of parties that can reveal the secret value. Let $\mathbb{Z}_N$ be the domain of values in the proposed scheme with a size $N = 2^l$, where l is the bit length. For an integer $d \geqslant 2$, we define $\mathbb{Z}_N^d$ as a d-dimension vector space over $\mathbb{Z}_N$.

The scheme of vector space secret sharing comprises the following three stages:

• **Setup.** Let $(1, 0, \ldots, 0)$ be a d-dimension target vector. The distributor defines the function $\boldsymbol{\psi}$ such that the public vector $\boldsymbol{\psi}(P_0), \boldsymbol{\psi}(P_1), \ldots, \boldsymbol{\psi}(P_{n-1})$ corresponding to each party satisfies the following: $(1, 0, \ldots, 0) \in \langle \boldsymbol{\psi}(P_j) : P_j \in A \rangle \Leftrightarrow A \in \mathit{\Lambda}$, where $A = \left\{P_{i_0}, P_{i_1}, \ldots, P_{i_{m-1}}\right\}$ is the minimal authorized subset, and m denotes the count of parties in A. For $0 \leqslant j < n$, $\boldsymbol{\psi}(P_j) \in \mathbb{Z}_N^d$ is the public d-dimensional vector of P_j.

• **Sharing.** For the secret $\sigma \in \mathbb{Z}_N$, the distributor P_i first selects a vector $\boldsymbol{w} = (w_0, w_1, \ldots, w_{d-1})^{\mathrm{T}} \in \mathbb{Z}_N^d$, where $w_0 = \sigma$ and $w_1, \ldots, w_n$ are random values over $\mathbb{Z}_N$. Subsequently, the distributor calculates $\boldsymbol{\psi}(P_j) \cdot \boldsymbol{w}$ and sends it to P_j $(0 \leqslant j < n, j \neq i)$ over a secure channel as P_j's share, that is, $\langle x \rangle_j = \boldsymbol{\psi}(P_j) \cdot \boldsymbol{w}$.

• **Reconstruction.** For any authorized subset A, since $(1, 0, \ldots, 0) \in \langle \boldsymbol{\psi}(P_j) : P_j \in A \rangle \Leftrightarrow A \in \mathit{\Lambda}$, there exists public contants $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ that are not all zeros, such that: $(1, 0, \ldots, 0) = \alpha_0 \cdot \boldsymbol{\psi}(P_{i_0}) + \cdots + \alpha_{m-1} \cdot \boldsymbol{\psi}(P_{i_{m-1}})$. In this case, the secret value $\sigma = w_0 = (1, 0, \ldots, 0) \cdot \boldsymbol{w} = (\alpha_0 \cdot \boldsymbol{\psi}(P_{i_0}) + \cdots + \alpha_{m-1} \cdot \boldsymbol{\psi}(P_{i_{m-1}})) \cdot \boldsymbol{w} = \alpha_0 \cdot \langle \sigma \rangle_0 + \alpha_1 \cdot \langle \sigma \rangle_1 + \cdots + \alpha_{m-1} \cdot \langle \sigma \rangle_{m-1}$. Therefore, we can reconstruct the secret s by multiplying each share $\langle \sigma \rangle$ with the corresponding constant, subsequently summing the calculated values.

3.2.2 *Function secret sharing*

FSS extends the concept of additive secret sharing. It diverges from classical secret sharing since the shared secret is not an element over a group, ring or field, but rather a computational function. Given a function $f : \{0,1\}^\kappa \rightarrow \mathbb{Z}_N$, each party P_i holds a share $[f]_i$, and these shares need to satisfy the following equation: $f(x) = [f(x)]_0 + \cdots + [f(x)]_n \bmod N$. More formally, an FSS scheme comprises two fundamental algorithms:

• $\mathsf{Gen}(1^\lambda, f) \rightarrow (k_0, \ldots, k_n)$ is used to generate secret keys, given the security parameter λ and a function f, and it outputs several keys $(k_0, \ldots, k_n)$.

• $\mathsf{Eval}(i, k_i, x) \rightarrow [f(x)]_i$ is used for evaluation, given the party index $i \in \{0, \ldots, n\}$, the key k_i and the public function input x; and it outputs the share $[f(x)]_i$ for each party respectively, where $f(x) = \sum_{i=0}^n [f(x)]_i$.

A distributed point function (DPF) is known as a concrete instance of (two-party) FSS. As stated in [31], a DPF is defined as follows:

$$f^{\cdot}_{a,b}(x) = \begin{cases} b, & \text{if } x = a, \\ 0^{|b|}, & \text{otherwise}, \end{cases} \tag{2}$$

where $a \in \{0,1\}^\kappa$ and $b \in \mathbb{Z}_N$.

Referring to [32–34], we discover that the FSS scheme for a DPF, that includes a set of algorithms $(\mathsf{Gen}^{\cdot}_{a,b}, \mathsf{Eval}^{\cdot}_{a,b})$, tends to meet both correctness and security demands.

Correctness. For any DPF $f^{\cdot}_{a,b}(x)$ and public input x, if $\mathsf{Gen}(1^\lambda, f) \rightarrow (k_0, k_1)$, then $\Pr[\mathsf{Eval}(0, k_0, x) + \mathsf{Eval}(1, k_1, x) = f^{\cdot}_{a,b}(x)] = 1$.

Security. In simple terms, the security of a DPF lies in the fact that even if an adversary obtains either k_0 or k_1, they still cannot learn the information of a and b.

Inspired by the scheme for a DPF, Boyle et al. [34] developed an efficient FSS scheme for a distributed comparison function (DCF). Similar to a DPF, a DCF is defined as follows:

$$f^{<}_{a,b}(x) = \begin{cases} b, & \text{if } x < a, \\ 0^{|b|}, & \text{otherwise}. \end{cases} \tag{3}$$

The FSS scheme designed for a DCF also involves two algorithms $(\mathsf{Gen}^{<}_{a,b}, \mathsf{Eval}^{<}_{a,b})$. Moreover, it also satisfies the aforementioned requirements of correctness and security.

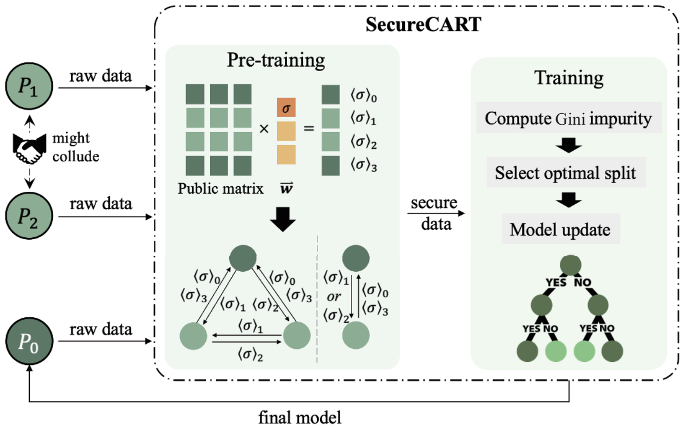

Figure 2 (Color online) System model of SecureCART.

4 System architecture

4.1 System model

Following the framework pMPL proposed in [5], the set of all parties is represented as $\mathcal{P} = \{P_0, P_1, P_2\}$, where P_0 is the privileged party, that will continually engage in the training process and ultimately acquire the training results. P_1 and P_2 serve as assistant parties, with the condition that either party can drop out or for both parties can collude with each other. Consequently, the minimal access structure is defined as $\Lambda = \{\{P_0, P_1, P_2\}, \{P_0, P_1\}, \{P_0, P_2\}\}$.

In a synchronous network, parties communicate via pairwise secure channels to participate in the training. However, to prevent private raw data from being exposed to other parties, each party performs vector space secret sharing ($\langle\cdot\rangle$-sharing) on its data before the training process. This implies that the communication data during the training process takes the form of $\langle\cdot\rangle$-shares.

For the sake of clarity, we consider the entire system in terms of two phases: pre-training and training, as depicted in Figure 2.

4.1.1 *Pre-training*

In this phase, parties primarily engage in the secret sharing of their data nd simultaneously execute necessary preprocessing tasks, such as precomputing vector multiplication triplets.

To execute vector space secret sharing, according to the properties of vector space secret sharing, a 4×3 public matrix is selected for pMPL, that can be regarded as consisting of four 3-dimensional public vector $\boldsymbol{\psi}(i)$, where $\boldsymbol{\psi}(i)$ represents the i-row of the public matrix $\boldsymbol{\psi}(\mathcal{P})$.

As mentioned in Subsection 3.2, the 4×3 public matrix $\boldsymbol{\psi}(\mathcal{P})$ should satisfy the following:

$$\begin{aligned}(1,0,0) &= \alpha_0 \cdot \boldsymbol{\psi}(0) + \alpha_1 \cdot \boldsymbol{\psi}(1) + \alpha_2 \cdot \boldsymbol{\psi}(2) \\ &= \alpha_0' \cdot \boldsymbol{\psi}(0) + \alpha_1' \cdot \boldsymbol{\psi}(1) + \alpha_3' \cdot \boldsymbol{\psi}(3) \\ &= \alpha_0'' \cdot \boldsymbol{\psi}(0) + \alpha_2'' \cdot \boldsymbol{\psi}(2) + \alpha_3'' \cdot \boldsymbol{\psi}(3).\end{aligned} \tag{4}$$

Note that constants $\alpha_0, \alpha_1, \ldots, \alpha_3''$ are supposed to be elements of $\mathbb{Z}_N$ and are public to all parties.

For the raw data σ to be shared, its owner P_i constructs a 3-dimensional vector $\boldsymbol{w} = (\sigma, w_1, w_2)^{\mathrm{T}}$ for it, where w_1 and w_2 are random numbers. Subsequently, the owner computes $\langle\sigma\rangle_j = \boldsymbol{\psi}(P_j) \cdot \boldsymbol{w}$, where $0 \leqslant j \leqslant 3$. After distributing shares, the privileged party P_0 is required to hold $\langle\sigma\rangle_0$ and $\langle\sigma\rangle_3$, while assistant parties P_1 and P_2 are required to hold $\langle\sigma\rangle_1$ and $\langle\sigma\rangle_2$, respectively.

The sharing process mentioned above has been formalized in protocol Π_{shr} [5], where parties execute $\Pi_{\mathrm{shr}}(P_i, \sigma)$ on all of the data required to be transferred to others.

4.1.2 *Training*

In this phase, the inputs of parties are in the form of $\langle\cdot\rangle$-shares. During the training, all of the intermediate results are also presented as $\langle\cdot\rangle$-shares to avoid leakage. Upon completion of training, only the privileged party possesses the final result, that is, a well-trained CART tree. A more detailed description of the training process is provided in Section 6.

Besides, the training phase may involve the reconstruction of shares. In the reconstruction protocol $\Pi_{\rm rec}(\mathcal{P}, \langle\sigma\rangle)$ [5], if no assistant party drops out, we can reveal the secret value via (5a). If an assistant party P_2 (or P_1) drops out, the secret value can be revealed using (5b) (or (5c)).

$$\begin{aligned}\sigma &= \alpha_0 \cdot \langle\sigma\rangle_0 + \alpha_1 \cdot \langle\sigma\rangle_1 + \alpha_2 \cdot \langle\sigma\rangle_2 && \text{(5a)}\\ &= \alpha'_0 \cdot \langle\sigma\rangle_0 + \alpha'_1 \cdot \langle\sigma\rangle_1 + \alpha'_3 \cdot \langle\sigma\rangle_3 && \text{(5b)}\\ &= \alpha''_0 \cdot \langle\sigma\rangle_0 + \alpha''_2 \cdot \langle\sigma\rangle_2 + \alpha''_3 \cdot \langle\sigma\rangle_3. && \text{(5c)}\end{aligned}$$

4.2 Threat model

As in the pMPL framework, the semi-honest (also referred as passive or honest-but-curious) security model is adopted in our framework, in which the adversary attempts to extract more information than expected during the interaction that occurs during the training process while strictly adhering to the protocol specifications. Moreover, we assume that the privileged party does not collude with any other party, whereas assistant parties have the potential for colluding with each other.

Formally, we establish the privacy of our scheme by employing the standard simulation paradigm with the real-ideal world model [35]. Assuming the presence of a P.P.T (probabilistic polynomial time) adversary $\mathcal{A}$ in the real world that can corrupt parties in two cases: corrupting P_0, or corrupting P_1 and P_2 (i.e., $P1$ and P_2 collude with each other). In the ideal world, we define $\mathcal{S}$ as a simulator, where $\mathcal{S}$ serves as an honest party and simulates the behavior of the adversary in the real world. Based on the ideal functionality $\mathcal{F}$, we establish the security of the semi-honest protocols Π by demonstrating the computational indistinguishability (denoted as $\cong$) between the views in the ideal and real worlds as follows:

$$\left\{\mathbf{view}^{\Pi}_{\mathcal{A}}(\lambda, \langle x\rangle_0, \langle x\rangle_1, \langle x\rangle_2), \mathrm{output}^{\Pi}(y)\right\} \cong \mathrm{Sim}(\lambda, \langle x\rangle_i, f(\langle x\rangle_0, \langle x\rangle_1, \langle x\rangle_2)).$$

4.3 Design goals

In this paper, we aim to enhance the efficiency and privacy protection of a multi-party decision tree training scheme with a privileged party. In summary, our design goals can be described as follows.

• **Privacy protection.** To ensure the privacy of private data among multiple parties, the inputs of parties and the intermediate results must not be improperly learned. Furthermore, given the uneven distribution of privileges among parties, access to the final model is granted only to the privileged party.

• **Robustness.** In this system, training should smoothly and correctly proceed to completion even in the event of dropout from one of the assistant parties or collusion between multiple assistant parties.

• **High efficiency.** The protocols used in this system should consume as little computing time and communication overhead as possible to accommodate some resource-constrained execution environments.

5 Building blocks

In this section, we augment the existing pMPL framework with novel sub-protocols, that is, secure division protocol, secure comparison protocol, and the extension protocol of secure multiplication. These sub-protocols will play a significant role in privacy protection for decision tree training within a multi-party setting.

Note that the secure addition protocol $\Pi_{\rm add}$, secure multiplication protocol $\Pi_{\rm mul}$, vector multiplication triplets generation protocol $\Pi_{\rm vmtgen}$, and sharing conversion protocols $\Pi_{\rm a2v}$, $\Pi_{\rm v2a}$ are already provided in [5]. Hence, we do not discuss them in this paper.

5.1 Secure division

Since division is one of the most commonly used operations, implementing a secure and efficient multi-party division protocol under the pMPL framework is essential. We provide the details of the secure division protocol $\Pi_{\rm div}$ in Algorithm 1.

The entire protocol comprises two phases: the preprocessing phase and the online phase, where the operations in the preprocessing phase are conducted offline. During the preprocessing phase, each party

Algorithm 1 Secure division $\Pi_{\text{div}}(\mathcal{P}, \langle x\rangle, \langle y\rangle)$

```
Preprocessing:
1: $P_i$ generates a random number $r_i$, for $i \in \{0,1,2\}$;
2: $P_i$ executes $\Pi_{\text{shr}}(P_i, r_i)$, for $i \in \{0,1,2\}$;
3: $P_i$ locally computes $\langle r\rangle_i = \langle r_0\rangle_i + \langle r_1\rangle_i + \langle r_2\rangle_i$, for $i \in \{0,1,2\}$. In addition, $P_0$ computes the alternate share $\langle r\rangle_3$ in the same way;
Input: $\langle x\rangle$ and $\langle y\rangle$;
Output: $\langle z\rangle = \langle \frac{x}{y}\rangle$;
1: Parties execute $\Pi_{\text{mul}}(\mathcal{P}, \langle r\rangle, \langle y\rangle)$ interactively to obtain $\langle r\cdot y\rangle$;
2: Parties execute $\Pi_{\text{rec}}(\mathcal{P}, \langle r\cdot y\rangle)$ interactively, and thus they all obtain $r\cdot y$;
3: Parties execute $\Pi_{\text{mul}}(\mathcal{P}, \langle r\rangle, \langle x\rangle)$ interactively to obtain $\langle r\cdot x\rangle$;
4: $P_i$ locally computes $\langle z\rangle_i = \frac{\langle r\cdot x\rangle_i}{r\cdot y}$ for $i \in \{0,1,2\}$, and $P_0$ computes an alternate share $\langle z\rangle_3 = \frac{\langle r\cdot x\rangle_3}{r\cdot y}$ additionally.
```

Algorithm 2 Secure comparison $\Pi_{\text{comp3}}(\mathcal{P}, \langle x\rangle, \langle y\rangle)$

```
Preprocessing:
1: $P_0$ evaluates $\mathsf{Gen}^{<}_{\frac{N}{2},1} \to (k_1, k_2)$, and sends $k_i$ to $P_i$ for $i \in \{1,2\}$;
Input: $\langle x\rangle$ and $\langle y\rangle$;
Output: $\langle z\rangle = \langle \mathbf{1}\{x > y\}\rangle$;
1: $P_0$ generates a random number $r$;
2: $P_i$ computes $\langle x-y\rangle_i = \langle x\rangle_i - \langle y\rangle_i$ locally, for $i \in \{0,1,2\}$;
3: Parties interactively execute $\Pi_{\text{extmul}}(\mathcal{P}, r, \langle x-y\rangle)$ to obtain $\langle r\cdot(x-y)\rangle$;
4: $P_0$ sends $\langle r\cdot(x-y)\rangle_0$ to $P_i$, who then exchange shares with each other, for $i \in \{1,2\}$. Thus, $P_1$ and $P_2$ compute (5a) locally to obtain $r\cdot(x-y)$;
5: $P_i$ evaluates $\mathsf{Eval}^{<}_{\frac{N}{2},1}(i, k_i, r\cdot(x-y)) \to [f]_i$, for $i \in \{1,2\}$;
6: $P_1$ and $P_2$ interactively reconstruct $f = [f]_1 + [f]_2$, and then one of them executes $\Pi_{\text{shr}}(P_i, f)$;
7: $P_i$ sets $\langle z\rangle_i = \langle f\rangle_i$, for $i \in \{1,2\}$. Besides, $P_0$ computes $\langle z\rangle_0 = \langle f\rangle_0 \oplus \mathsf{MSB}(r)$ locally and sets $\langle z\rangle_3 = \langle f\rangle_3$.
```

P_i first generates a random number $r_i \in \mathbb{Z}_N$ respectively and executes $\Pi_{\text{shr}}(P_i, r_i)$ on it. Consequently, P_i obtains three shares $\langle r_0\rangle_i$, $\langle r_1\rangle_i$ and $\langle r_2\rangle_i$. P_0 additionally holds $\langle r_0\rangle_3$, $\langle r_1\rangle_3$ and $\langle r_2\rangle_3$. Subsequently, P_i sums up these shares locally to obtain $\langle r\rangle_i = \langle r_0\rangle_i + \langle r_1\rangle_i + \langle r_2\rangle_i$, and P_0 also calculates an alternate share $\langle r\rangle_3$. Thus, parties cooperate to acquire the corresponding share $\langle r\rangle_i$ without knowing the real value of r; therefore, r can be used later to mask the value that requires to be kept secret.

During the online phase, parties execute $\Pi_{\text{mul}}(\mathcal{P}, \langle r\rangle, \langle y\rangle)$ jointly. Thus, P_i obtains $\langle r\cdot y\rangle_i$. Next, P_0 obtains an alternate share $\langle r\cdot y\rangle_3$. Subsequently, by executing the reconstruction protocol $\Pi_{\text{rec}}(\mathcal{P}, \langle r\cdot y\rangle)$, parties collaborate to reveal the value of $r\cdot y$, that is, all of the parties obtain the denominator y masked by the prepared random value r. Analogously, parties interactively execute $\Pi_{\text{mul}}(\mathcal{P}, \langle r\rangle, \langle x\rangle)$. After that, P_i holds $\langle r\cdot x\rangle_i$, and P_0 additionally holds an alternate share $\langle r\cdot x\rangle_3$. Finally, each party P_i computes $\langle z\rangle_i = \frac{\langle r\cdot x\rangle_i}{r\cdot y}$ locally. Besides, P_0 extra computes $\langle z\rangle_3 = \frac{\langle r\cdot x\rangle_3}{r\cdot y}$.

Theorem 1 (Correctness of Π_{div})**.** For shares $\langle x\rangle$ and $\langle y\rangle$ held by online parties, where $\langle x\rangle, \langle y\rangle \in \mathbb{Z}_N$, Π_{div} can correctly outputs shares of division result $\langle \frac{x}{y}\rangle$ for all parties. Proof details are provided in Appendix A.

Theorem 2 (Security of Π_{div})**.** The division protocol Π_{div} securely realizes the functionality $\mathcal{F}_{\text{div}}$ in the case of passive adversaries. Proof details are provided in Appendix B.

5.2 Secure comparison

During the training process of the decision tree, comparison operations are required to determine the magnitude of the Gini impurity. As we know, FSS-based protocols are generally more effective in two prevalent scenarios: (1) offline communication is more cost-effective than online communication, or (2) network latency is the primary factor leading to performance degradation. In an MPC environment with preprocessing, FSS-based methods outperform mixed-mode MPC approaches in terms of both round complexity and online round communication.

Thus, we design an FSS-based comparison protocol, that inputs the shares of $\langle x\rangle$ and $\langle y\rangle$ as input, and outputs the secret shared comparison result $\langle z\rangle$, where $z = 1$ if $x > y$ and $z = 0$ otherwise. Note that for integers over the ring $\mathbb{Z}_N$, we map the positive numbers to $\{0, 1, \ldots, N/2-1\}$, and the negative numbers are mapped to $\{N/2, \ldots, N-1\}$. In other words, the most significant bit of x is 0, that is, $\mathsf{MSB}(x) = 0$, when $x \in \mathbb{Z}_N$ is positive and $\mathsf{MSB}(x) = 1$ otherwise.

For the case where no party drops out, the secure three-party comparison protocol Π_{comp3} is described in Algorithm 2. In the preprocessing phase, the privileged party P_0 generates keys k_i for the assistant parties

Algorithm 3 Secure comparison $\Pi_{\text{comp2}}(\mathcal{P}, \langle x\rangle, \langle y\rangle)$

```
Input: $\langle x\rangle$ and $\langle y\rangle$;
Output: $\langle z\rangle = \langle \mathbf{1}\{x > y\}\rangle$;
1: $P_i$ generates a random number $r_i$, for $i \in \{0,1\}$;
2: $P_0$ and $P_1$ interactively evaluate $\mathsf{Gen}^{<}_{r_0+r_1,1} \to (k_0, k_1)$, where $P_i$ holds $k_i$ for $i \in \{0,1\}$;
3: $P_i$ locally computes $\langle y\rangle_i - \langle x\rangle_i + r_i/\alpha'_i$ for $i \in \{0,1,2\}$. $P_0$ additionally computes $\langle y\rangle_3 - \langle x\rangle_3$;
4: Parties jointly reconstruct $y - x + r_0 + r_1$ using (5b);
5: $P_i$ evaluates $\mathsf{Eval}^{<}_{r_0+r_1,1}(i, k_i, y - x + r_0 + r_1) \to [f]_i$, for $i \in \{0,1\}$;
6: $P_i$ executes $\Pi_{\text{shr}}(P_i, [f]_i)$ respectively, for $i \in \{0,1\}$;
7: $P_i$ locally computes $\langle z\rangle_i = \langle [f]_0\rangle_i + \langle [f]_1\rangle_i$ for $i \in \{1,2\}$. Besides, $P_0$ computes $\langle z\rangle_3 = \langle [f]_0\rangle_3 + \langle [f]_1\rangle_3$.
```

by $\mathsf{Gen}^{<}_{\frac{N}{2},1}$, where $i \in \{1,2\}$. In the online phase, P_0 generates a random number r. Each party P_i locally computes $\langle x\rangle_i - \langle y\rangle_i$ to obtain $\langle x - y\rangle_i$, and P_0 additionally compute $\langle x - y\rangle_3 = \langle x\rangle_3 - \langle y\rangle_3$. Subsequently, all parties jointly execute $\Pi_{\text{extmul}}(\mathcal{P}, r, \langle x - y\rangle)$, and thus P_0 obtains $\langle r \cdot (x - y)\rangle_0$ and $\langle r \cdot (x - y)\rangle_3$, while the assistant party P_i obtains $\langle r \cdot (x - y)\rangle_i$, where $i \in \{1,2\}$. The assistant parties receive $\langle r \cdot (x - y)\rangle_0$ from P_0 and exchange their shares with each other; consequently, they can reconstruct $r \cdot (x - y)$. Since the value of r is unknown to P_1 and P_2, and P_0 has not reconstructed $r \cdot (x - y)$ although it knows r, none of the parties will have the knowledge of the real value of $x - y$. P_i evaluates $\mathsf{Eval}^{<}_{\frac{N}{2},1}(i, k_i, r \cdot (x - y))$ to obtain the additive shares of $\mathbf{1}\{r \cdot (x - y) < \frac{N}{2}\}$, that are denoted as $[f]_i$ for $i \in \{1,2\}$. Subsequently, P_1 and P_2 sum their shares to obtain the value of f, secretly sharing f subsequently. Therefore, P_i holds $\langle z\rangle_i = \langle f\rangle_i$ for $i \in \{1,2\}$. Additionally, P_0 holds $\langle z\rangle_3 = \langle f\rangle_3$ and locally computes $\langle z\rangle_0 = \langle f\rangle_0 \oplus \mathsf{MSB}(r)$.

For the case where one of the assistant parties drops out, for example, if P_2 drops out, the secure two-party comparison protocol Π_{comp2} is described in Algorithm 3. Our protocol shares similarities in terms of ideation with the comparison protocol in [6], with the difference that our protocol refers to the approach in [34] to eliminate the dependency on a trusted dealer.

Theorem 3 (Correctness of Π_{comp})**.** For shares $\langle x\rangle$ and $\langle y\rangle$ held by online parties, where $\langle x\rangle, \langle y\rangle \in \mathbb{Z}_N$, Π_{comp} can correctly output shares of the comparison result for all parties.

Proof. To prove the correctness of Π_{comp}, it is necessary to demonstrate that the shares outputted by Π_{comp} can be reconstructed using Π_{rec} to obtain the correct value of $\mathbf{1}\{x > y\}$. The reconstruction process is explained for two scenarios as follows:

If no assistant party drops out,

$$\begin{aligned} z &= \alpha_0 \cdot \langle z\rangle_0 + \alpha_1 \cdot \langle z\rangle_1 + \alpha_2 \cdot \langle z\rangle_2 \\ &= \alpha_0 \cdot (\mathsf{MSB}(r) \oplus \langle f\rangle_0) + \alpha_1 \cdot \langle f\rangle_1 + \alpha_2 \cdot \langle f\rangle_2 \\ &= \mathsf{MSB}(r) \oplus f \\ &= \begin{cases} \mathsf{MSB}(r) \oplus 1, & \text{if } r \cdot (x - y) < \dfrac{N}{2}, \\ \mathsf{MSB}(r) \oplus 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{6}$$

More specifically, there is $\mathsf{MSB}(x - y) = 0$ when $x > y$. If $\mathsf{MSB}(r) = 0$, then $r \cdot (x - y) < N/2$; therefore, parties have $z = 0 \oplus 1 = 1$. If $\mathsf{MSB}(r) = 1$, then $r \cdot (x - y) \geqslant N/2$; therefore, parties can obtain $z = 1 \oplus 0 = 1$. Regarding $x < y$, $\mathsf{MSB}(x - y) = 1$. If $\mathsf{MSB}(r) = 0$, then $r \cdot (x - y) \geqslant N/2$; therefore, parties obtain $z = 0 \oplus 0 = 0$. If $\mathsf{MSB}(r) = 1$, then $r \cdot (x - y) < N/2$; therefore, parties can obtain $z = 1 \oplus 1 = 0$. To summarize, parties can reconstruct $z = 1$ when $x > y$ and $z = 0$ when $x \leqslant y$, that satisfies $z = \mathbf{1}\{x > y\}$.

If one of the assistant parties (for example, P_2) drops out,

$$\begin{aligned} z &= \alpha'_0 \cdot \langle z\rangle_0 + \alpha'_1 \cdot \langle z\rangle_1 + \alpha'_3 \cdot \langle z\rangle_3 \\ &= \alpha'_i \cdot (\langle [f]_0\rangle_i + \langle [f]_1\rangle_i),\ i \in \{0,1,3\} \\ &= [f]_0 + [f]_1 \\ &= \begin{cases} 1, & \text{if } y - x + r_0 + r_1 < r_0 + r_1, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{7}$$

Thus, the online parties can reconstruct $z = 1$ when $x > y$ and $z = 0$ otherwise, that is, $z = \mathbf{1}\{x > y\}$.

Theorem 4 (Security of Π_{comp})**.** The comparison protocol Π_{comp} securely realizes the functionality $\mathcal{F}_{\text{comp}}$ in the case of passive adversaries. Proof details are provided in Appendix C.

Algorithm 4 Extension of multiplication $\Pi_{\text{extmul}}(\mathcal{P}, x, \langle y \rangle)$

```
Preprocessing:
 1: Parties execute $\Pi_{\text{vmtgen}}(\mathcal{P})$ to generate the shares of vector multiplication triplets $\langle u \rangle$, $\langle v \rangle$ and $\langle h \rangle$;
Input: $x$ (held by $P_0$) and $\langle y \rangle$;
Output: $\langle z \rangle = \langle x \cdot y \rangle$;
  - If no assistant party drops out:
 1:     $P_0$ receives $\langle u \rangle_i$ from $P_i$, where $i \in \{1, 2\}$, and then obtains $u$ by computing (5a);
  - If one assistant party ($P_2$) drops out:
 2:     $P_0$ receives $\langle u \rangle_1$ from $P_1$, and then obtains $u$ by computing (5b);
 3: $P_0$ locally computes $e = x + u$ and sends $e$ to $P_i$ for $i \in \{1, 2\}$;
 4: $P_i$ locally compute $\langle d \rangle_i = \langle y \rangle_i + \langle v \rangle_i$ for $i \in \{0, 1, 2\}$, Besides, $P_0$ computes $\langle d \rangle_3 = \langle y \rangle_3 + \langle v \rangle_3$;
 5: Parties jointly execute $\Pi_{\text{rec}}(\mathcal{P}, \langle d \rangle)$ and thus they all obtain $d = y + v$;
 6: $P_i$ locally computes $\langle z \rangle_i = \langle h \rangle_i - \langle v \rangle_i \cdot e$ for $i \in \{1, 2\}$. Meanwhile, $P_0$ locally computes $\langle z \rangle_3 = \langle h \rangle_3 - \langle v \rangle_3 \cdot e$ and $\langle z \rangle_0 = \frac{x \cdot d}{\alpha_0} + \langle h \rangle_0 - \langle v \rangle_0 \cdot e$.
```

5.3 Extension of secure multiplication

The extension protocol of secure multiplication is used to perform multiplication in the case that the privileged party P_0 owns the plaintext data x, and $\langle y \rangle$ is secret-shared among three parties, and they jointly compute the shares of result $\langle x \cdot y \rangle$. Through this protocol, the data held exclusively by the privileged party can be securely multiplied with the intermediate data in the form of secret sharing, without the need for an additional execution of the sharing protocol. We describe the extension protocol of secure multiplication in Algorithm 4.

Similar to Π_{mul} [5], the vector multiplication triplets need to be generated by executing Π_{vmtgen} during preprocessing. During the online phase, P_0 first receives the share(s) from the assistant party (or parties), and reconstructs u. Subsequently, P_0 masks the plaintext x with u, which gives $e = x + u$, and then sends e to the assistant parties. All parties locally compute $\langle d \rangle = \langle y \rangle + \langle v \rangle$, and then reconstruct $d = y + v$ by executing Π_{rec}. Considering that $x \cdot y = x \cdot (y + v) - v \cdot (x + u) + u \cdot v$ and that x is held only by P_0, we let P_0 compute $\langle z \rangle_0 = \frac{x \cdot d}{\alpha_0} + \langle h \rangle_0 - \langle v \rangle_0 \cdot e$, while the other shares are computed as $\langle z \rangle_i = \langle h \rangle_i - \langle v \rangle_i \cdot e$ for $i \in \{1, 2, 3\}$.

Theorem 5 (Correctness of Π_{extmul})**.** For the plaintext x held by P_0 and the shares $\langle y \rangle$ held by online parties, where $x, \langle y \rangle \in \mathbb{Z}_N$, Π_{extmul} can correctly outputs shares of the multiplication result $\langle x \cdot y \rangle$ for all parties. Proof details are provided in Appendix D.

Theorem 6 (Security of Π_{extmul})**.** The extension protocol of secure multiplication Π_{extmul} securely realizes the functionality $\mathcal{F}_{\text{extmul}}$ in the case of passive adversaries. Proof details are provided in Appendix E.

6 SecureCART

In this section, we outline the process of designing SecureCART, a scheme for training decision trees with privacy preservation under the pMPL framework by combining building blocks.

6.1 Overview

Similar to [6], the training samples in SecureCART are partitioned vertically among three participants, indicating that P_i holds all the sample data corresponding to certain features, rather than holding all the feature data for certain samples (known as horizontal partitioning). However, unlike the common multi-party decision tree training models previously seen, the privilege of the three parties in our training model is asymmetrical. P_0, as the privileged party, is regarded as the initiator of the training, that holds not only the samples of certain features but also the labels, and ultimately obtains the final training result. The two assistant parties P_1 and P_2 serve as databases, supplementing P_0 with additional feature data of the existing samples, with the possibility of dropping out.

Based on the above structure, during the pre-training phase, each party locally represents its samples belonging to different splits of features as vectors. Additionally, the privileged party P_0 represents samples as vectors based on distinct labels. Subsequently, P_0 completes initialization, and the assistant parties P_1 and P_2 execute the sharing protocol on their vectorized sample data.

After completing the local preprocessing of samples, parties jointly determine the best split of features and update the model using a combination of building blocks, according to the original CART training

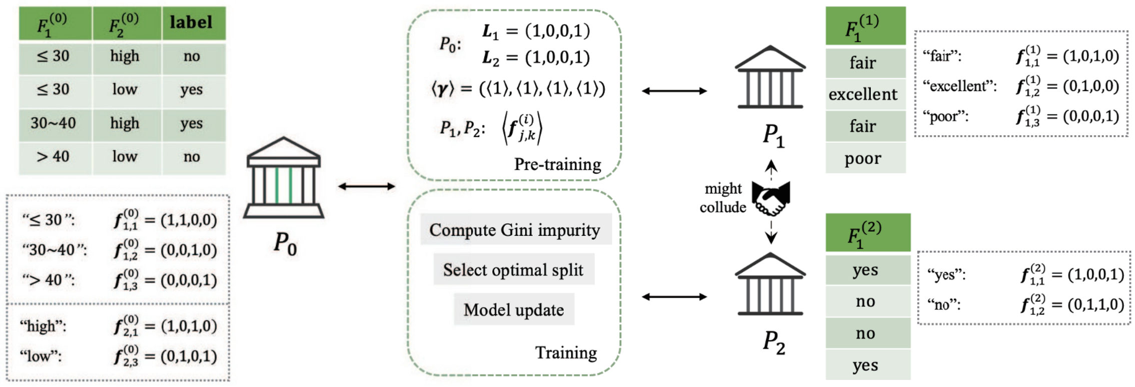
$F_1^{(0)}$ $F_2^{(0)}$ label
≤ 30 high no
≤ 30 low yes
30~40 high yes
> 40 low no
"≤ 30": $f_{1,1}^{(0)} = (1,1,0,0)$
"30~40": $f_{1,2}^{(0)} = (0,0,1,0)$
"> 40": $f_{1,3}^{(0)} = (0,0,0,1)$
"high": $f_{2,1}^{(0)} = (1,0,1,0)$
"low": $f_{2,3}^{(0)} = (0,1,0,1)$
P_0
P_0: $L_1 = (1,0,0,1)$ $L_2 = (1,0,0,1)$
$\langle\gamma\rangle = (\langle 1\rangle, \langle 1\rangle, \langle 1\rangle, \langle 1\rangle)$
P_1, P_2: $\langle f_{j,k}^{(i)}\rangle$
Pre-training
Compute Gini impurity
Select optimal split
Model update
Training
P_1
might collude
P_2
$F_1^{(1)}$ fair excellent fair poor
"fair": $f_{1,1}^{(1)} = (1,0,1,0)$
"excellent": $f_{1,2}^{(1)} = (0,1,0,0)$
"poor": $f_{1,3}^{(1)} = (0,0,0,1)$
$F_1^{(2)}$ yes no no yes
"yes": $f_{1,1}^{(2)} = (1,0,0,1)$
"no": $f_{1,2}^{(2)} = (0,1,1,0)$

Figure 3 (Color online) Instance of SecureCART.

algorithm and the formula for the variant form of the Gini impurity. The above training process will be completed when the stopping conditions are met, yielding the well-trained model of CART tree to P_0.

6.2 Detailed scheme

In this subsection, we offer a detailed explanation of SecureCART, the proposed privacy-preserving decision tree training scheme with a privileged party, that comprises two phases, namely, pre-training and training, where training involves the calculation of the variant Gini impurity, selection of the optimal split, and model updating.

The set of all features held by three parties is denoted as $\mathcal{F} = \{F_1^{(0)}, F_2^{(0)} \ldots, F_1^{(1)}, \ldots, F_1^{(2)}\}$, where $F_j^{(i)}$ represents the dataset corresponding to the j-th feature held by P_i, with $j \in \{1, \ldots, m^{(i)}\}$. Additionally, $F_j^{(i)}$ can be divided into $\delta_j^{(i)}$ splits. Each split is represented in vector form as $\boldsymbol{f}_{j,k}^{(i)}$, that is a vector indicating whether the sample belongs to the k-th split of $F_j^{(i)}$, with $k \in \{1, \ldots, \delta_j^{(i)}\}$. It has to be mentioned that all of the vectors used in SecureCART have the same length t, which is the number of training samples. In these vectors, 1 indicates belonging (or equivalence); conversely, 0 represents non-belonging (or non-equivalence).

We consider Figure 3 as a simple example. In this scenario, the privileged party P_0 possesses the private datasets of samples $F_1^{(0)}$ and $F_2^{(0)}$, along with the corresponding labels. The two assistant parties, P_1 and P_2, own their sensitive data $F_1^{(1)}$ and $F_1^{(2)}$ respectively. Note that we assume that the datasets held by different parties have already been matched through some method, such as private set intersection.

Local preprocessing of samples (pre-training). Each party represents each of its datasets as $\boldsymbol{f}_{j,k}^{(i)}$ in a vector form based on distinct splits of features, and P_0 organizes the samples into M classes based on labels, representing the samples belonging to the μ-th class as a vector $\boldsymbol{L}_\mu$, where $\mu \in \{1, 2, \ldots, M\}$. Subsequently, all parties apply the sharing protocol to some of their plaintext data that needs to be shared. P_0 initializes the vector of current available samples $\boldsymbol{\gamma} = (1, 1, \ldots, 1)$, which is then secret-shared by executing $\Pi_{\text{shr}}(P_0, \boldsymbol{\gamma})$. Due to the potential for assistant parties to drop out, if their datasets are only held by themselves, the training process will be interrupted when one of them drops out. Consequently, it becomes essential for P_1 and P_2 to secret-share their private data, ensuring the continuity of training in the event of a dropout. Therefore, P_1 and P_2 respectively execute $\Pi_{\text{shr}}(P_i, \boldsymbol{f}_{j,k}^{(i)})$, for $i \in \{1, 2\}$, to protect their plaintext data from being disclosed to others during the training process. Regarding the privileged party P_0, which never drops out, it does not need to share its raw data and can participate in the training via executing Π_{extmul}.

After implementing the above steps, each assistant party $P_{i'}$ holds $\langle \boldsymbol{f}_{j,k}^{(i)} \rangle_{i'}$ and $\langle \boldsymbol{\gamma} \rangle_{i'}$, where $i' \in \{1, 2\}$, while P_0 holds $\boldsymbol{f}_{j,k}^{(0)}$, $\langle \boldsymbol{\gamma} \rangle_0$ and $\langle \boldsymbol{\gamma} \rangle_3$. These shares serve as the inputs for the first round of training; this indicates that preprocessing only needs to be performed once at the very beginning and does not need to be involved in subsequent recursion.

Calculation of the variant Gini impurity. First, it is necessary to determine whether the current node meets the conditions to be a leaf node. If it does meet the condition, the corresponding label should be returned. Next, the variant Gini values for various feature splits need to be calculated. According to (1), parties interactively compute $\langle \boldsymbol{C}_\mu \rangle = \boldsymbol{L}_\mu \circ \langle \boldsymbol{\gamma} \rangle$ by executing Π_{extmul} element-wise. And then parties

Algorithm 5 SecureCART

```
Input: $\langle \boldsymbol{f}^{(i)}_{j,k} \rangle$, $\langle \boldsymbol{C}_\mu \rangle$ and $\langle \boldsymbol{\gamma} \rangle$.
Output: A well-trained CART tree.
 1: if the conditions for being a leaf node are satisfied then
 2:     return The label with majority class;
 3: end if
 4: for $\mu \leftarrow 1, \ldots, M$ do
 5:     Parties jointly execute $\langle \boldsymbol{C}_\mu \rangle \leftarrow \Pi_{\mathrm{extmul}}(\mathcal{P}, \boldsymbol{L}_\mu, \langle \boldsymbol{\gamma} \rangle)$ in an element-wise manner;
 6: end for
 7: Initialize $\langle \tilde{g}^* \rangle$, $\langle i^* \rangle$, $\langle j^* \rangle$ and $\langle k^* \rangle$ to $\langle 0 \rangle$;
 8: for $i \leftarrow 0, 1, 2$ do
 9:     for $j \leftarrow 1, \ldots, m^{(i)}$ do
10:         for $k \leftarrow 1, \ldots, \delta^{(i)}_j$ do
11:             - If $i = 0$, parties jointly executes $\langle d_l \rangle \leftarrow \Pi_{\mathrm{extmul}}(\mathcal{P}, \boldsymbol{f}^{(i)}_{j,k}, \langle \boldsymbol{\gamma} \rangle)$ and $\langle d_r \rangle \leftarrow \Pi_{\mathrm{extmul}}(\mathcal{P}, \mathbf{1} - \boldsymbol{f}^{(i)}_{j,k}, \langle \boldsymbol{\gamma} \rangle)$;
12:             - If $i \in \{1, 2\}$, parties jointly executes $\langle d_l \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle \boldsymbol{f}^{(i)}_{j,k} \rangle, \langle \boldsymbol{\gamma} \rangle)$ and $\langle d_r \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle \mathbf{1} - \boldsymbol{f}^{(i)}_{j,k} \rangle, \langle \boldsymbol{\gamma} \rangle)$;
13:             Set $\langle lc \rangle = \langle 0 \rangle$ and $\langle rc \rangle = \langle 0 \rangle$;
14:             for $\mu \leftarrow 1, \ldots, M$ do
15:                 - If $i = 0$, parties jointly executes $\langle lc_\mu \rangle \leftarrow \Pi_{\mathrm{extmul}}(\mathcal{P}, \boldsymbol{f}^{(i)}_{j.k}, \langle \boldsymbol{C}_\mu \rangle)$ and $\langle rc_\mu \rangle \leftarrow \Pi_{\mathrm{extmul}}(\mathcal{P}, \mathbf{1} - \boldsymbol{f}^{(i)}_{j.k}, \langle \boldsymbol{C}_\mu \rangle)$;
16:                 - If $i \in \{1, 2\}$, Parties jointly executes $\langle lc_\mu \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle \boldsymbol{f}^{(i)}_{j,k} \rangle, \langle \boldsymbol{C}_\mu \rangle)$ and $\langle rc_\mu \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle \mathbf{1} - \boldsymbol{f}^{(i)}_{j,k} \rangle, \langle \boldsymbol{C}_\mu \rangle)$;
17:                 Parties jointly executes $\langle lc^2_\mu \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle lc_\mu \rangle, \langle lc_\mu \rangle)$ and $\langle rc^2_\mu \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle rc_\mu \rangle, \langle rc_\mu \rangle)$;
18:                 Parties locally compute $\langle lc \rangle = \langle lc \rangle + \langle lc^2_\mu \rangle$ and $\langle rc \rangle = \langle rc \rangle + \langle rc^2_\mu \rangle$;
19:             end for
20:             Parties jointly executes $\langle lp \rangle \leftarrow \Pi_{\mathrm{div}}(\mathcal{P}, \langle lc \rangle, \langle d_l \rangle)$ and $\langle rp \rangle \leftarrow \Pi_{\mathrm{div}}(\mathcal{P}, \langle rc \rangle, \langle d_r \rangle)$;
21:             Parties locally compute $\langle \tilde{g}^{(i)}_{j,k} \rangle = \langle lp \rangle + \langle rp \rangle$;
22:         end for
23:     end for
24:     for $j \leftarrow 1, \ldots, m^{(i)}$ do
25:         for $k \leftarrow 1, \ldots, \delta^{(i)}_j$ do
26:             Parties jointly executes $\langle u \rangle \leftarrow \Pi_{\mathrm{comp}}(\mathcal{P}, \langle \tilde{g}^{(i)}_{j,k} \rangle, \langle \tilde{g}^* \rangle)$;
27:             Parties jointly executes $\langle \tilde{g}^* \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle u \rangle, \langle \tilde{g}^{(i)}_{j,k} - \tilde{g}^* \rangle) + \langle \tilde{g}^* \rangle$, and compute the corresponding $\langle k^* \rangle$, $\langle j^* \rangle$
                and $\langle i^* \rangle$ in the same way;
28:         end for
29:     end for
30: end for
31: $P_0$ receives $\langle k^* \rangle_{i'}$, $\langle j^* \rangle_{i'}$ and $\langle i^* \rangle_{i'}$ from $P_{i'}$, where $i' \in \{1, 2\}$, and uses (5a) to obtain the optimal feature $F^{(i^*)}_{j^*}$ and
    its corresponding best split (i.e., the $k^*$-th split);
32: Parties execute $\langle \boldsymbol{\gamma}_l \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle \boldsymbol{\gamma} \rangle, \langle \boldsymbol{f}^{(i^*)}_{j^*.k^*} \rangle)$ and $\langle \boldsymbol{\gamma}_r \rangle \leftarrow \Pi_{\mathrm{mul}}(\mathcal{P}, \langle \boldsymbol{\gamma} \rangle, \langle \mathbf{1} - \boldsymbol{f}^{(i^*)}_{j^*.k^*} \rangle)$ in an element-wise manner.
```

compute the value of the variant Gini impurity for each split of features, denoted as $\langle \tilde{g}^{(i)}_{j,k} \rangle$, by interactively executing Π_{mul}, Π_{extmul} and Π_{div}, along with local addition operations. Since the computational and communication overhead of Π_{div} is higher than that of the local addition operations, unlike [6], we minimize the overhead by first summing and ultimately dividing once.

Selection of the optimal split. Parties interactively execute Π_{comp} to select the maximum value of the variant Gini impurity, denoted as $\langle \tilde{g}^* \rangle$. The best feature split corresponding to $\langle \tilde{g}^* \rangle$ is $\langle \boldsymbol{f}^{(i^*)}_{j^*,k^*} \rangle$. The assistant parties send their shares $\langle k^* \rangle_{i'}$, $\langle j^* \rangle_{i'}$ and $\langle i^* \rangle_{i'}$, where $i' \in \{1, 2\}$, to the privileged party P_0. Thus, only P_0 has the ability able to reconstruct the value of i^*, j^* and k^*, and learn the optimal feature $F^{(i)}_{j,k}$ and its best split (i.e., the k-th split).

Model update. In this step, the set of available samples $\langle \boldsymbol{\gamma} \rangle$ for the current node is supposed to be divided into two subsets according to the optimal feature split $\langle \boldsymbol{f}^{(i^*)}_{j^*,k^*} \rangle$. Specifically, parties interactively compute $\langle \boldsymbol{\gamma}_l \rangle = \langle \boldsymbol{\gamma} \rangle \circ \langle \boldsymbol{f}^{i^*}_{j^*,k^*} \rangle$ by executing $\Pi_{\mathrm{mul}}(\mathcal{P}, \langle \boldsymbol{\gamma} \rangle, \langle \boldsymbol{f}^{(i^*)}_{j^*.k^*} \rangle)$ in an element-wise manner, and then locally compute $\langle \boldsymbol{\gamma}_r \rangle = \langle \boldsymbol{\gamma} \rangle - \langle \boldsymbol{\gamma}_l \rangle$, with $\langle \boldsymbol{\gamma}_l \rangle$ as the dataset available for the left child node and $\langle \boldsymbol{\gamma}_r \rangle$ as the dataset available for the right child node.

The above operations for one node with the set of available samples $\langle \boldsymbol{\gamma} \rangle$ are summarized in Algorithm 5, that are recursively executed.

7 Evaluation

In this section, we present the experimental setup of SecureCART and the selected datasets, followed by its performance evaluation and comparison with recent similar schemes.

Table 2 Accuracy and efficiency of SecureCART

Dataset	Accuracy (%)		Runtime on LAN (s)		Runtime on WAN (s)	
	Original CART	SecureCART	PriVDT	SecureCART	PriVDT	SecureCART
Iris	97.82	97.24	0.88	0.41	43.24	35.72
Bank Marketing	88.16	87.02	10.78	3.29	238.70	173.20
Credit Card Client	83.84	83.17	13.42	4.73	311.28	221.24

7.1 Experimental setup

Settings. We performed all experiments using the existing pMPL framework[1] on three Linux servers equipped with 2.3 GHz Intel(R) Xeon(R) Processor (Skylake, IBRS) and 8 GB of RAM running the 64-bit Ubuntu 18.04 system, each of which represents one of P_0, P_1 and P_2. Besides, the FSS-based comparison protocol is implemented based upon the LibFSS library[2]. We simulate two kinds of network environments: one is the LAN setting with a bandwidth of 2 Gbps and the RTT (round-trip time) latency is 0.3 ms, the other one is the WAN setting with 40 Mbps bandwidth and 40 ms RTT latency. In both network environments, we implement SecureCART in C++ over the ring $\mathbb{Z}_{2^l}$, with a setting of $l = 64$ and the least significant bits $l_f = 20$ as the fractional part, which aligns with the settings of PriVDT [6] and pMPL [5]. Furthermore, considering the parameters given in [5], the public matrix can be set as $\psi(\mathcal{P}) = [\psi(0), \psi(1), \psi(2), \psi(3)]^{\mathrm{T}}$, where $\psi(0) = (1, 0, 1)$, $\psi(1) = (1, 1, 2^l - 1)$, $\psi(2) = (2, 2, 2^l - 3)$ and $\psi(3) = (3, 3, 2^l - 4)$. Then we can obtain the value of the constants using (4), where $\alpha_0 = 1$, $\alpha_1 = 2^l - 2$, $\alpha_2 = 1$, $\alpha'_0 = 1$, $\alpha'_1 = 2^l - 3$, $\alpha'_3 = 1$, $\alpha''_0 = 1$, $\alpha''_2 = 3$ and $\alpha''_3 = 2^l - 2$.

Datasets. Here we utilize the same datasets as that in [6], which are provided by the UC Irvine Machine Learning Repository [36].

• **Iris Dataset.** It is among the earliest known datasets for classification tasks. It comprises instances labeled into three categories, with each label corresponding to a distinct kind of iris plant and containing 50 instances. And each instance has four numeric features.

• **Bank Marketing Dataset.** It contains 4521 instances with 17 features, and the final feature serves as the desired result, which predicts if the client will subscribe to a fixed deposit, in the form of a binary ("Yes" or "No").

• **Default of Credit Card Client Dataset.** It contains 30000 instances with 23 features, and uses the default payment as the output, which is in a binary form with "Yes =1" and "No =0".

7.2 Evaluation of accuracy

We analyze the accuracy of models trained by the original CART and SecureCART on various datasets respectively; the results of this analysis are provided in Table 2. Although the accuracy of SecureCART is slightly reduced as compared to other schemes, which is mainly caused by the accuracy sacrifice of the division protocol and the fixed-point representation, this minor loss is acceptable in practical scenarios.

7.3 Evaluation of efficiency

In this subsection, we first evaluate the performance of building blocks. Table 3 gives the runtime and communication overhead for the existing secure interactive protocols of pMPL in the LAN and WAN environments respectively, which is not covered in [5]. Here, 3PC indicates the scenario where no party drops out, and 2PC denotes a scenario wherein one assistant party drops out. In practical experiments, we enhance the efficiency of the scheme by executing protocols on vectors, where the dimension of the vector corresponds to the number of instances in the dataset. We can observe that the runtime of each protocol increases as the vector dimension gradually increases, but this increase is slow rather than in a multiplicative form. Moreover, the addition and multiplication protocols we used are faster even compared to the two-party PriVDT [6].

For the proposed secure division protocol Π_{div}, Figure 4(a) illustrates the runtime and communication overhead of the division operation with PriVDT. We can observe that Π_{div} of SecureCART is approximately 93.5×–560.4× faster than PriVDT. Furthermore, as the vector dimension increases, the difference of runtime between these division protocols gradually widens. In the case of 3PC, Π_{div} of SecureCART

1) https://github.com/FudanMPL/pMPL.

2) https://github.com/frankw2/libfss.

Table 3 Runtime and communication overhead of secure sharing, reconstruction, addition and multiplication protocols

Setting	Dimension	Runtime on LAN (ms)				Runtime on WAN (ms)				Communication (kB)			
		Shr.	Rec.	Add.	Mul.	Shr.	Rec.	Add.	Mul.	Shr.	Rec.	Add.	Mul.
3PC	100	0.08	0.08	<0.001	0.16	2.04	40.40	<0.001	79.79	2.34	3.12	0.00	6.25
	500	0.14	0.10	<0.001	0.21	2.18	40.61	<0.001	80.24	11.71	15.62	0.00	31.25
	1000	0.19	0.12	<0.001	0.25	2.28	40.86	<0.001	81.86	23.43	31.24	0.00	62.50
	1500	0.26	0.15	<0.001	0.43	2.39	41.22	<0.001	82.67	35.13	46.86	0.00	93.75
	2000	0.33	0.17	<0.001	0.54	2.56	41.53	<0.001	83.40	46.86	62.48	0.00	120.50
2PC	100	0.08	0.04	<0.001	0.10	2.04	40.28	<0.001	79.55	2.34	1.56	0.00	3.12
	500	0.14	0.05	<0.001	0.13	2.18	40.48	<0.001	79.98	11.71	7.81	0.00	15.62
	1000	0.19	0.06	<0.001	0.17	2.28	40.62	<0.001	81.62	23.43	15.63	0.00	31.24
	1500	0.26	0.07	<0.001	0.35	2.39	40.87	<0.001	82.43	35.13	23.43	0.00	46.86
	2000	0.33	0.09	<0.001	0.46	2.56	41.12	<0.001	83.16	46.86	31.25	0.00	62.48

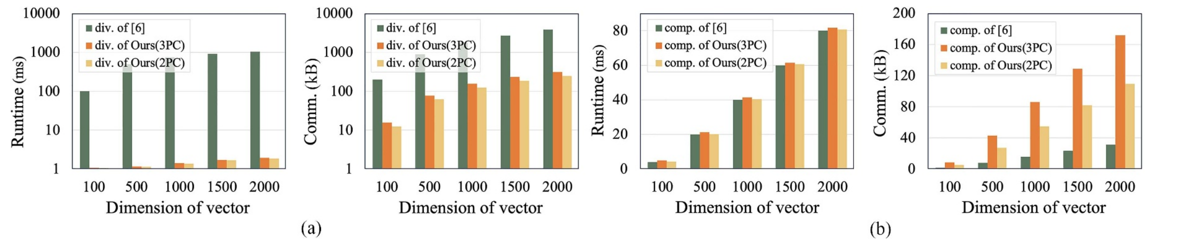

Figure 4 (Color online) Comparing the runtime and communication overhead of (a) the division operation and (b) the comparison operation with that for PriVDT.

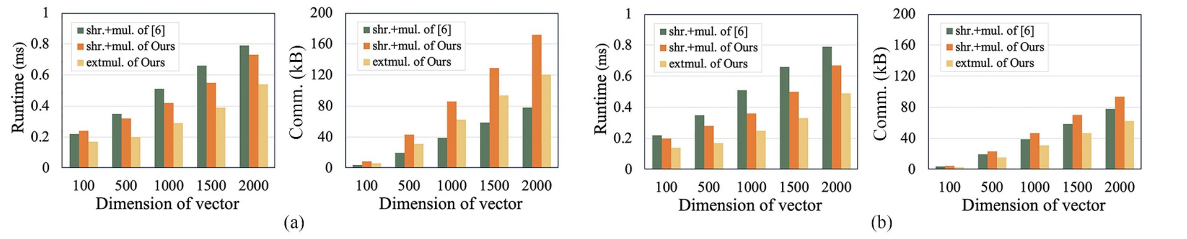

Figure 5 (Color online) Comparing the runtime and communication overhead of the multiplication operation with that for PriVDT. (a) Under the 3PC setting of SecureCART; (b) under the 2PC setting of SecureCART.

reduces the communication overhead by 91% compared to that of PriVDT, while in the case of 2PC, our Π_{div} incurred the communication overhead of only 7% as compared to that for PriVDT.

For the secure comparison protocol Π_{comp} of SecureCART, Figure 4(b) shows the runtime and communication overhead of the comparison operation with that of PriVDT. Since our scheme is based on the special asymmetric framework of pMPL, that is, it includes two roles, that of a privileged party and assistant party, the FSS-based Π_{comp} designed for SecureCART is inevitably more complex than those for the two-party PriVDT. Nevertheless, our division protocol Π_{comp}, whether in the 2PC or 3PC setting, has a runtime that is roughly comparable to that for PriVDT's. This is because the time overhead of the comparison protocols mainly results from the element-wise $\mathsf{Eval}^{<}$ in the vector, and the runtime of other operations considered in our comparison, such as Π_{extmul} and Π_{shr}, is relatively small as compared to this value.

In the 2PC setting for our comparison protocol, there is a significant increase in the communication overhead as compared to that for PriVDT due to the conversion from $[\cdot]$-sharing to $\langle\cdot\rangle$-sharing completed by Π_{shr} and the masking process adapted to the pMPL framework through Π_{extmul}. In the 3PC setting, the involvement of more participants leads to a further increase in the communication overhead of our comparison protocol. Although the performance of our Π_{comp} is inferior to that of PriVDT, it remains within an acceptable range, and it still outperforms comparison protocols of schemes such as [28]. In future studies, we aim to improve the multi-party FSS to achieve a better performance of our secure comparison protocol for pMPL.

For the proposed extension of secure multiplication protocol Π_{extmul}, we compare its runtime and communication overhead with that of the original multiplication operation of pMPL and PriVDT in Figure 5. In the case where the privileged party has the plaintext data x, and each party holds a share $\langle y\rangle$, Π_{extmul} can perform secure multiplication without secret-sharing x. In contrast, PriVDT and pMPL require executing the secure multiplication protocol after Π_{shr}. Therefore, it is reasonable to compare our

Π_{extmul} with the existing $\Pi_{\text{shr}} + \Pi_{\text{mul}}$. In the 3PC setting of SecureCART, Π_{extmul} is around $1.1\times$–$1.42\times$ faster than $\Pi_{\text{shr}} + \Pi_{\text{mul}}$ of PriVDT and $1.25\times$–$1.37\times$ faster than $\Pi_{\text{shr}} + \Pi_{\text{mul}}$ of pMPL. However, our communication overhead is higher than PriVDT due to the greater number of parties. In the 2PC setting of SecureCART, Π_{extmul} is approximately $1.5\times$ faster than PriVDT and $1.26\times$–$1.39\times$ faster than pMPL. Additionally, the communication overhead of Π_{extmul} is reduced by 20.2% compared to PriVDT.

We then evaluate the performance of SecureCART on different datasets. Table 2 provides the runtime of SecureCART on LAN and WAN, comparing it with the training procedure of PriVDT. Our scheme is over $2\times$ faster than PriVDT on LAN for three kinds of datasets, and over $1.3\times$ faster than on WAN. The better performance on LAN is due to the higher communication overhead of 3PC compared to the two-party PriVDT. This implies that SecureCART's advantage is even greater in a lower latency environment.

8 Conclusion

In this paper, we propose SecureCART, a privacy-preserving decision tree training scheme based on a special framework, pMPL. To be specific, we design three novel building blocks based on pMPL, including the secure division, the secure comparison and the extension of secure multiplication, which achieve better performance compared to existing solutions via random masking and FSS. SecureCART, which is completed by combining these building blocks, consequently provides significant performance advantages. Moreover, while SecureCART in this paper is conducted by three parties, we can also incorporate more assistant parties into the training by designing suitable public matrices.

In future studies, we will continue to optimize building blocks that need to be improved and expand SecureCART to make it applicable to various forms of datasets. Additionally, since the privacy-preserving decision tree inference is not include in our scheme, and similar solutions all have shortcomings in privacy or efficiency, we also consider designing a multi-party privacy-preserving decision tree inference scheme based on pMPL, which will prove to be secure and efficient.

Acknowledgements This work was supported in part by Major Program (JD) of Hubei Province (Grant No. 2023BAA027), National Natural Science Foundation of China (Grant Nos. 62202339, 62172307, U21A20466, 62325209), New 20 Project of Higher Education of Jinan (Grant No. 202228017), and Fundamental Research Funds for the Central Universities (Grant No. 2042023KF0203).

Supporting information Appendixes A–E. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Lu S, Zheng J, Cao Z, et al. A survey on cryptographic techniques for protecting big data security: present and forthcoming. Sci China Inf Sci, 2022, 65: 201301
2 An Y, Meng H, Gao Y, et al. Application of machine learning method in optical molecular imaging: a review. Sci China Inf Sci, 2020, 63: 111101
3 Liu F, Zheng Z, Shi Y, et al. A survey on federated learning: a perspective from multi-party computation. Front Comput Sci, 2024, 18: 181336
4 Sun G. New progress in research and application of machine learning. Chin J Electron, 2020, 29: 991
5 Song L, Wang J, Wang Z, et al. pMPL: a robust multi-party learning framework with a privileged party. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2022. 2689–2703
6 Chen H, Li H, Wang Y, et al. PriVDT: an efficient two-party cryptographic framework for vertical decision trees. IEEE Trans Inform Forensic Secur, 2023, 18: 1006–1021
7 Hao M, Li H, Chen H, et al. FastSecNet: An efficient cryptographic framework for private neural network inference. IEEE Trans Inform Forensic Secur, 2023, 18: 2569–2582
8 Quinlan J R. Induction of decision trees. Mach Learn, 1986, 1: 81–106
9 Quinlan J R. C4.5: Programs for Machine Learning. Amsterdam: Elsevier, 2014
10 Lewis R J. An introduction to classification and regression tree (cart) analysis. In: Proceedings of Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, 2000
11 Lindell Y, Pinkas B. Privacy preserving data mining. In: Proceedings of Annual International Cryptology Conference, 2000. 36–54
12 Xiao M J, Huang L S, Luo Y L, et al. Privacy preserving ID3 algorithm over horizontally partitioned data. In: Proceedings of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05), 2005. 239–243
13 Samet S, Miri A. Privacy preserving ID3 using gini index over horizontally partitioned data. In: Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, 2008. 645–651
14 Hao M, Li H, Xu G, et al. Efficient, private and robust federated learning. In: Proceedings of Annual Computer Security Applications Conference, 2021. 45–60
15 Li A, Zhang L, Tan J, et al. Sample-level data selection for federated learning. In: Proceedings of IEEE INFOCOM 2021-IEEE Conference on Computer Communications, 2021. 1–10

16 Ma Q, Deng P. Secure multi-party protocols for privacy preserving data mining. In: Proceedings of the 3rd International Conference on Wireless Algorithms, Systems, and Applications, 2008. 526–537
17 Khodaparast F, Sheikhalishahi M, Haghighi H, et al. Privacy preserving random decision tree classification over horizontally and vertically partitioned data. In: Proceedings of the 16th International Conference on Dependable, Autonomic and Secure Computing, 2018. 600–607
18 Liu L, Chen R, Liu X, et al. Towards practical privacy-preserving decision tree training and evaluation in the cloud. IEEE Trans Inform Forensic Secur, 2020, 15: 2914–2929
19 Du W L, Zhan Z J. Building decision tree classifier on private data. In: Proceedings of the IEEE International Conference on Privacy, Security and Data Mining, 2002
20 She R, Wang K, Xu Y, et al. Pushing feature selection ahead of join. In: Proceedings of the SIAM International Conference on Data Mining, 2005. 536–540
21 Wang K, Xu Y, Yu P S, et al. Building decision trees on records linked through key references. In: Proceedings of the SIAM International Conference on Data Mining, 2005. 576–580
22 Vaidya J, Clifton C. Privacy-preserving decision trees over vertically partitioned data. In: Proceedings of IFIP Annual Conference on Data and Applications Security and Privacy, 2005. 139–152
23 Vaidya J, Clifton C, Kantarcioglu M, et al. Privacy-preserving decision trees over vertically partitioned data. ACM Trans Knowl Discov Data, 2008, 2: 1–27
24 Dansana J, Dey D, Kumar R. A novel approach: cart algorithm for vertically partitioned database in multi-party environment. In: Proceedings of IEEE Conference on Information & Communication Technologies, 2013. 829–834
25 Hu Y, Niu D, Yang J, et al. FDML: a collaborative machine learning framework for distributed features. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019. 2232–2240
26 Cheng K, Fan T, Jin Y, et al. SecureBoost: a lossless federated learning framework. IEEE Intell Syst, 2021, 36: 87–98
27 Abspoel M, Escudero D, Volgushev N. Secure training of decision trees with continuous attributes. Proc Privacy Enhancing Technol, 2021, 2021: 167–187
28 Wu Y, Cai S, Xiao X, et al. Privacy preserving vertical federated learning for tree-based models. 2020. ArXiv:2008.06170
29 Zheng Y, Xu S, Wang S, et al. Privet: a privacy-preserving vertical federated learning service for gradient boosted decision tables. IEEE Trans Serv Comput, 2023, 16: 3604–3620
30 Brickell E F. Some ideal secret sharing schemes. In: Proceedings of Workshop on the Theory and Application of Cryptographic Techniques, 1989. 468–475
31 Gilboa N, Ishai Y. Distributed point functions and their applications. In: Proceedings of the 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, 2014. 640–658
32 Boyle E, Gilboa N, Ishai Y. Function secret sharing. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2015. 337–367
33 Boyle E, Gilboa N, Ishai Y. Function secret sharing: improvements and extensions. In: Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2016. 1292–1303
34 Boyle E, Chandran N, Gilboa N, et al. Function secret sharing for mixed-mode and fixed-point secure computation. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2021. 871–900
35 Canetti R. Security and composition of multiparty cryptographic protocols. J Cryptology, 2000, 13: 143–202
36 Bache K, Lichman M. UCI machine learning repository. 2013. https://archive.ics.uci.edu