Efficient and Privacy-preserving Outsourced Training of Decision Tree Models Based on (Leveled) Fully Homomorphic Encryption

Tongyu Xu[2⋆], Jun Wang[2⋆⋆], Honglian Liang[2], and Shiwei Xu[1,2⋆⋆⋆]

[1] Engineering Research Center of Intelligent Technology for Agriculture, Ministry of Education, Huazhong Agricultural University, Wuhan, China
[2] College of Informatics, Huazhong Agricultural University, Wuhan, China

Abstract. Training machine learning models is computationally intensive, making cloud-based outsourcing an attractive solution to alleviate local resource constraints. However, untrusted cloud environments pose serious privacy risks to both training data and resulting models. Existing works primarily rely on multi-party computation (MPC) or lattice-based Homomorphic Encryption (HE), which often incur high communication or computation overheads. To address these challenges, we propose an efficient privacy-preserving scheme for outsourced decision tree training. Specifically, we leverage Symmetric Homomorphic Encryption (SHE) to achieve faster training speed. However, since SHE only supports integer-based homomorphic operations, we propose a Modified Gini Impurity Index (MGII) to adapt to this restriction and use Single Instruction Multiple Data (SIMD) packing to accelerate processing. Experimental results demonstrate that our scheme significantly reduces overall execution time compared to related works and achieves comparable (and for deeper trees, better) accuracy, while security analysis confirms that data and model confidentiality are preserved.

Keywords: Decision Tree · Outsourced Training · (Leveled) Fully Homomorphic Encryption · Symmetric Homomorphic Encryption.

1 Introduction

Machine learning (ML) models have become fundamental to a wide range of intelligent applications. However, training these models is computationally intensive and often exceeds the capabilities of individual data owners. Outsourcing the training process to a cloud service provider has therefore emerged as a practical solution. Nevertheless, directly outsourcing plaintext data introduces serious privacy risks to both the training dataset and the resulting model parameters. To address these concerns, data owners typically employ secure computation techniques to protect sensitive data throughout the outsourced training process.

⋆ Tongyu Xu and Jun Wang contributed equally to this work.
⋆⋆ Tongyu Xu and Jun Wang contributed equally to this work.
⋆⋆⋆ Corresponding author: xushiwei@mail.hzau.edu.cn

We consider the outsourced training of decision tree (DT) models. DTs are widely used for classification tasks and are commonly applied in privacy-sensitive domains such as financial risk assessment, medical diagnosis, and market analysis. Training a DT involves constructing a tree by selecting optimal feature-split pairs based on a labeled dataset.

Privacy-preserving decision tree techniques can be broadly divided into outsourced inference and outsourced training. Decision tree inference is relatively lightweight, as it only involves routing a data instance through a fixed tree structure. As a result, a large body of existing work [28, 7, 16, 18, 6, 25] has realized outsourced inference using multi-party computation (MPC) or fully homomorphic encryption (FHE).

In contrast, outsourced training is significantly more challenging. It requires extensive data-dependent computations and frequent non-linear branching to determine optimal split points. Existing training approaches [4, 20, 22, 8, 19, 5, 24] primarily rely on MPC or approximate FHE schemes such as CKKS, leading to high communication overhead, heavy computational cost, or potential accuracy loss due to approximation. To address these limitations, we adopt Symmetric Homomorphic Encryption (SHE) [23] to enable efficient and accurate outsourced training of decision tree models.

Our main contributions are summarized as follows:

- We propose an interactive protocol based on SHE for outsourced decision tree training, providing strong privacy protection for both the training data and model parameters.
- To accommodate that SHE only supports integer-based homomorphic operations, we propose a Modified Gini Impurity Index (MGII) that eliminates division operations to better suit its homomorphic computation, and we also employ SIMD packing in the protocol to accelerate processing.
- We conduct extensive experiments to evaluate the performance of our scheme. The results show that, compared to related works, our scheme achieves comparable or better model accuracy, particularly for deeper trees, while significantly reducing computation and overall execution time in realistic network environments. In addition, our approach exhibits predictable linear scalability and supports efficient training on large-scale datasets.

2 Prior Works

In privacy-preserving decision tree research, existing work can be broadly divided into outsourced inference and outsourced training. Outsourced inference involves routing encrypted data instances through a fixed tree structure and is therefore computationally lightweight. As a result, a large body of work [28, 7, 16, 18, 6, 25] has explored this setting using MPC or HE. For example, some studies [28] adopt MPC techniques, while others [7, 16, 18] use TFHE [15] for efficient boolean gate evaluation, and [6, 25] employ Ring-LWE-based schemes such as BGV [10] or BFV.

In contrast, outsourced training is significantly more challenging. It requires extensive data-dependent computations, such as aggregating instance distributions, as well as frequent non-linear operations, such as comparing impurity values to determine optimal splits. As a result, efficient end-to-end solutions for outsourced decision tree training remain limited, and existing approaches typically involve trade-offs in efficiency, scalability, or accuracy.

One line of work [20, 4, 22, 8, 19] adopts secret-sharing-based MPC techniques to enable outsourced training. These protocols typically rely on multiple non-colluding servers, leading to substantial inter-server communication overhead and reduced efficiency compared to single-server settings.

Another line of work [5, 24] targets single-server settings using approximate FHE schemes such as CKKS [13]. Since CKKS primarily supports polynomial arithmetic, these methods rely on polynomial approximations to evaluate non-linear functions. This may introduce accuracy loss in the resulting decision tree models. In addition, whether operating in a leveled setting [5] or using bootstrapping [24], these approaches incur high computational overhead, limiting their practical efficiency.

Table 1 summarizes a qualitative comparison between our scheme and existing approaches in terms of cryptographic techniques, server assumptions, and computation and communication overhead.

Table 1: Related works comparison

Schemes	[20, 4, 22, 8, 19]	[7, 16, 18]	[25, 6]	[5, 24]	**Our Work**
Num of Servers	> 1	1	1	1	1
Techniques	MPC (secret sharing)	[12] TFHE [15]	BGV/BFV [10]	CKKS [13]	SHE [23]
P-PO Training	✓	✗	✗	✓	✓
Computation	Low	High	Medium	Medium	Low
Communication	Medium	High	Medium	Medium	Low

Note: P-PO stands for privacy-preserving outsourced.

3 Preliminaries

3.1 Decision Trees

We consider binary decision tree (BDT) models (e.g., CART [21]). Suppose the training dataset contains n instances, where each instance is a d-dimensional feature vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$ associated with a label $lab_{\boldsymbol{x}} \in [1, c]$. The training process starts from the root node. At each node N, the goal is to select an optimal feature-split pair (j, v_j) based on the assigned subset, where $j \in [1, d]$. The optimal split is chosen to minimize the Gini Impurity Index (GII). The dataset is then partitioned into two subsets, and the process continues until a stopping criterion (e.g., maximum depth) is met. Leaf nodes store the final classification results, while each internal node has exactly two children.

3.2 Symmetric Homomorphic Encryption (SHE)

SHE [23] is an integer-based (leveled) fully homomorphic encryption (FHE) technique that supports efficient homomorphic addition and multiplication. The SHE scheme comprises the following three fundamental algorithms:

- **Key Generation** $(sk, pp) \leftarrow \mathsf{SHE.KeyGen}(k_0, k_1, k_2)$: The key generation algorithm takes three security parameters k_0, k_1, k_2 (satisfying $k_1 \ll k_2 < k_0/2$) and outputs a secret key sk and a public parameter pp. Specifically, the secret key is constructed as $sk = (p, q, \mathcal{R})$, where p and q are two large randomly generated primes such that $|p| = |q| = k_0$, and $\mathcal{R}$ is a random integer meeting $|\mathcal{R}| = k_2$. The plaintext message space is defined as $\mathcal{M} = \{m \mid m \in [-2^{k_1-1}, 2^{k_1-1})\}$. Finally, the public modulus $\mathcal{N} = pq$ is computed, and the public parameter is globally set as $pp = (k_0, k_1, k_2, \mathcal{N})$.
- **Encryption** $[\![m]\!]_{sk} \leftarrow \mathsf{SHE.Enc}(m, sk)$: To encrypt a plaintext message $m \in \mathcal{M}$ under the secret key $sk = (p, q, \mathcal{R})$, the algorithm instances two random integers $r \in \{0,1\}^{k_2}$ and $r' \in \{0,1\}^{k_0}$. The corresponding ciphertext is then computed as $[\![m]\!]_{sk} = (r\mathcal{R} + m)(1 + r'p) \bmod \mathcal{N}$.
- **Decryption** $m \leftarrow \mathsf{SHE.Dec}([\![m]\!]_{sk}, sk)$: To decrypt a given ciphertext $[\![m]\!]_{sk}$ using the secret key sk, the algorithm deterministically recovers the original message by computing $m = ([\![m]\!]_{sk} \bmod p) \bmod \mathcal{R}$.

Given a plaintext message m_0 and two ciphertexts $[\![m_1]\!]_{sk}$ and $[\![m_2]\!]_{sk}$ encrypted under the same secret key sk, SHE supports the following homomorphic operations: $m_0 \oplus [\![m_1]\!]_{sk} = [\![m_0 + m_1]\!]_{sk}$, $m_0 \otimes [\![m_1]\!]_{sk} = [\![m_0 m_1]\!]_{sk}$, $[\![m_1]\!]_{sk} \oplus [\![m_2]\!]_{sk} = [\![m_1 + m_2]\!]_{sk}$ and $[\![m_1]\!]_{sk} \otimes [\![m_2]\!]_{sk} = [\![m_1 m_2]\!]_{sk}$, where $\oplus$ and $\otimes$ respectively represent the homomorphic addition and multiplication. Furthermore, while SHE supports an almost unlimited number of homomorphic additions, it restricts the maximum number of consecutive ciphertext multiplications to $\lfloor (k_0/2k_2) - 1 \rfloor$, which is characteristic of leveled FHE schemes.

While SHE is fundamentally designed as a symmetric-key algorithm, it can be transformed into an asymmetric public-key encryption scheme [26]. This is achieved by publishing a public key $pk = r_0 \otimes [\![0]\!]_{sk}$, where r_0 is a k_2-bit random number. Using pk, anyone can encrypt a message m as $[\![m]\!]_{pk} = m \oplus pk = m \oplus (r_0 \otimes [\![0]\!]_{sk}) = [\![m + (r_0 \times 0)]\!]_{sk} = [\![m]\!]_{sk}$. The resulting ciphertext $[\![m]\!]_{pk}$ can then be conventionally decrypted using the original secret key sk.

In order to improve the efficiency of homomorphic addition between ciphertexts, SHE supports the SIMD batching technique [9]. This mechanism allows multiple scalar plaintexts to be packed into distinct independent slots within a single ciphertext. Consequently, a single homomorphic addition executed on packed ciphertexts simultaneously applies the operation element-wise across all corresponding slots, significantly amortizing the computational cost.

4 Scheme Overview

4.1 System Architecture

Consistent with prevailing paradigms in privacy-preserving outsourced computing, our system architecture involves two primary entities namely the Data Owner (DO) and the Cloud Service Provider (CSP). The DO possesses a sensitive training dataset and wishes to train a decision tree model but lacks the requisite computational infrastructure. Conversely, the CSP is equipped with abundant computing resources and provides outsourced training services.

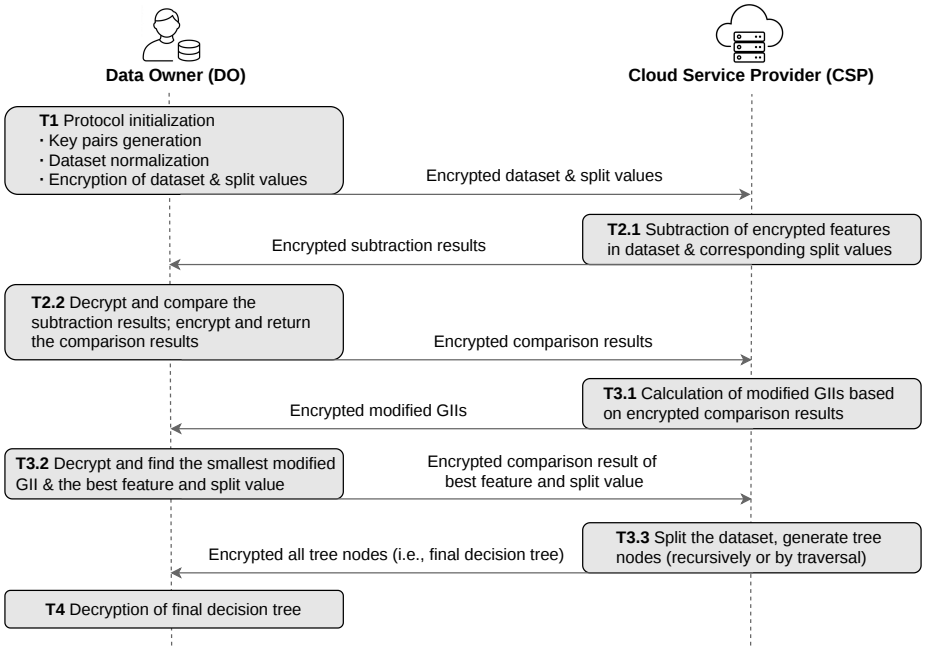


Fig. 1: Overview of outsourced training system architecture

An illustrative overview of our outsourced decision tree training protocol is depicted in Figure 1. The overall workflow comprises a local initialization step followed by a 3-round interactive protocol between the DO and the CSP, which are systematically structured into four distinct phases (T1 to T4).

- **Protocol Initialization (Phase T1):** Before the protocol commences, the DO locally performs key generation, dataset normalization, and data encoding. The DO then encrypts the normalized dataset and initial model parameters, transmitting only the ciphertexts and the public keys to the CSP.
- **Interactive Training (Phases T2 and T3):** The CSP undertakes the majority of the computationally intensive operations of the model training over

the encrypted data. However, because the underlying SHE scheme naturally supports only additions and multiplications (and not non-linear operations like comparisons), the CSP periodically engages the DO. In these brief interactions, the DO receives intermediate ciphertexts, performs lightweight decryptions and comparisons, re-encrypts the boolean results, and returns them to the CSP. This interactive mechanism effectively bypasses the need for the severe computational overheads associated with homomorphic comparison circuits.
- **Result Retrieval (Phase T4):** Upon completion of the training phase, the CSP executes the final interactive round (T4) by returning the entirely encrypted tree structure (including all internal and leaf nodes) to the DO, who then decrypts them to recover the final plaintext decision tree model.

4.2 Security Models and Desired Properties

In our threat model, we assume that the DO is fully trusted, whereas the CSP is honest-but-curious (also known as semi-honest). This means the CSP will faithfully execute the prescribed protocol but will attempt to infer sensitive information from the data, intermediate results, and messages received during the execution.

Under this threat model, our proposed scheme aims to guarantee the following two fundamental privacy properties:

- **Training Dataset Privacy**: The CSP must not be able to deduce any plaintext information regarding the original training instances or their corresponding labels provided by the DO.
- **DT Model Privacy**: Throughout the entire outsourced training process, the CSP must be prevented from acquiring any specifics of the model. This includes the selected features and split values at internal nodes, as well as the final classification results at leaf nodes.

To enforce these privacy guarantees, we employ the SHE algorithm as our core cryptographic primitive. SHE has been formally proven to satisfy IND-CPA (Indistinguishability under Chosen Plaintext Attack) security [27], ensuring that the CSP cannot distinguish between the encryptions of any two distinct plaintexts without the secret key.

5 Outsourced Training Protocol

In this section, we elaborate on the cryptographic and algorithmic details of our outsourced training protocol, corresponding to the interactive phases previously illustrated in Figure 1.

Algorithm 1 Protocol Initialization

Participating party: Data Owner DO
Input: Dataset features $feat$ and labels lab
Output: Encrypted packed features $[\![\boldsymbol{feat}]\!]_{sk_1}$, encrypted negated split value candidates $[\![-\boldsymbol{v}]\!]_{sk_1}$, and encrypted one-hot labels $[\![\boldsymbol{LAB}]\!]_{sk_2}$

Generate two SHE key pairs $(pk_1, sk_1), (pk_2, sk_2)$
Normalize all the features $feat$
Generate split value candidates v for all features
$\boldsymbol{feat} \leftarrow$ SIMD_encode($feat$) *// SIMD encoding of features*
$-\boldsymbol{v} \leftarrow$ SIMD_encode($-v$) *// SIMD encoding of negated split candidates*
$[\![\boldsymbol{feat}]\!]_{sk_1} \leftarrow$ SHE.Enc($\boldsymbol{feat}, sk_1$)
$[\![-\boldsymbol{v}]\!]_{sk_1} \leftarrow$ SHE.Enc($-\boldsymbol{v}, sk_1$)
$\boldsymbol{LAB} \leftarrow$ One_hot_encode(lab) *// Encode labels using one-hot encoding*
$[\![\boldsymbol{LAB}]\!]_{sk_2} \leftarrow$ SHE.Enc($\boldsymbol{LAB}, sk_2$)
return $[\![\boldsymbol{feat}]\!]_{sk_1}$, $[\![-\boldsymbol{v}]\!]_{sk_1}$, $[\![\boldsymbol{LAB}]\!]_{sk_2}$ *// Transmit to* CSP

5.1 Protocol Initialization (T1)

Before the interactive protocol commences, the DO locally executes the protocol initialization (Phase T1 in Figure 1). The complete procedure for this protocol initialization is summarized in Algorithm 1.

Phase T1. Let the training dataset of the DO comprise n training instances. The DO first generates two independent SHE key pairs denoted as (pk_1, sk_1) and (pk_2, sk_2). Because the underlying SHE scheme exclusively supports integer arithmetic, the DO normalizes each feature x_j (for $j \in [1, d]$) across all training data vectors into an appropriate integer range $[-2^{b-1}, 2^{b-1})$, with $b \in \mathbb{Z}^+$. Concurrently, the DO generates a predefined number of split value candidates, with total count denoted by svc (i.e., v_k for $k \in [1, svc]$), for all features and normalizes these candidates into the identical integer range. To facilitate the subsequent homomorphic subtractions between x_j and v_k, the DO encodes all candidates as negative values ($-v_k$) prior to encryption under sk_1.

To optimize the computational efficiency of encryption and homomorphic subtraction, the DO leverages the SIMD packing technique to pack multiple feature vectors $[x_{11}, x_{21}, ..., x_{d1}], ..., [x_{1n}, x_{2n}, ..., x_{dn}]$ into independent slots. These slots are then bundled into a single ciphertext, denoted as $[\![x_{11}, x_{21}, ..., x_{d1}, ..., x_{1n}, x_{2n}, ..., x_{dn}]\!]_{sk_1}$. For each split value candidate v_k, the DO packs and encrypts $-v_k$ in the same packed format, yielding $[\![-v_k, -v_k, ..., -v_k, ..., -v_k, -v_k, ..., -v_k]\!]_{sk_1}$ for $k \in [1, svc]$. During this packing process, each slot is exactly b-bits wide, and any two adjacent slots are separated by a zero-filled padding bit to prevent arithmetic carry bits from corrupting adjacent slots during homomorphic evaluations.

Furthermore, the DO encodes the categorical label $lab_{\boldsymbol{x}} \in [1, c]$ of each feature vector $\boldsymbol{x}$ into a one-hot vector representation. This vector comprises c elements with exactly one 1 and $(c-1)$ zeros, where the index of the 1 corresponds to the value of $lab_{\boldsymbol{x}}$. For instance, a one-hot vector $[0, 0, 1, 0]$ indicates a total of

$c = 4$ classes and specifies that the label of vector $\boldsymbol{x}$ is $lab_{\boldsymbol{x}} = 3$. Consequently, the aggregate one-hot vectors for all n training instances form a comprehensive one-hot label matrix $\boldsymbol{LAB}_{n\times c} = [lab_{l,i}]_{n\times c}$, where $lab_{l,i} \in \{0, 1\}$ for $1 \leq l \leq n$ and $1 \leq i \leq c$. Upon completing this encoding, the DO encrypts each element within $\boldsymbol{LAB}_{n\times c}$ using sk_2, producing the encrypted label matrix $[\![\boldsymbol{LAB}_{n\times c}]\!]_{sk_2}$.

Finally, the DO transmits all the aforementioned ciphertexts (the packed feature vectors, the negated split value candidates, and the one-hot label matrix) to the CSP, triggering the first interactive round of the protocol.

5.2 The First Round (T2)

The first interactive round corresponds to Phase T2 in Figure 1.

Phase T2.1. During this round, the CSP receives all the encrypted feature vectors, negated split value candidates, and one-hot labels from the DO. Since the feature vectors and split value candidates are aligned within SIMD slots, the CSP efficiently performs homomorphic additions (which corresponds to subtraction due to negated candidates) across all encrypted feature vectors. This yields the encrypted subtraction results $[\![\boldsymbol{sr}^{(k)}]\!]_{sk_1} = [\![x_{11}-v_k, x_{21}-v_k, ..., x_{d1}-v_k, ..., x_{1n}-v_k, x_{2n}-v_k, ..., x_{dn}-v_k]\!]_{sk_1}$ for $k \in [1, svc]$, which are subsequently transmitted back to the DO.

Phase T2.2. Upon receiving the ciphertexts, the DO executes Algorithm 2. Specifically, the DO decrypts all $[\![\boldsymbol{sr}^{(k)}]\!]_{sk_1}$ using sk_1 to recover the plaintext matrices $\boldsymbol{sr}^{(k)}$ and evaluates the mathematical sign of each subtraction result. If the result is positive, the DO sets the corresponding comparison bit cr to 1; otherwise, it sets it to 0. After evaluating each split value candidate v_k ($k \in [1, svc]$), the DO obtains a boolean comparison result vector $\boldsymbol{cr}^{(k)} = [cr_{11}^{(k)}, cr_{21}^{(k)}, ..., cr_{d1}^{(k)}, ..., cr_{1n}^{(k)}, cr_{2n}^{(k)}, ..., cr_{dn}^{(k)}]$, where $cr_{jl}^{(k)} \in \{0, 1\}$ for $j \in [1, d]$ and $l \in [1, n]$. Consequently, all svc comparison result vectors can be represented as a three-dimensional matrix $\boldsymbol{CR}_{svc\times d\times n} = [cr_{k,j,l}]_{svc\times d\times n}$, where $cr_{k,j,l} \in \{0, 1\}$ for $1 \leq k \leq svc$, $1 \leq j \leq d$, and $1 \leq l \leq n$.

Algorithm 2 Secure Comparison Algorithm

Participating party: Data Owner DO
Input: Encrypted subtraction results $[\![\boldsymbol{sr}^{(k)}]\!]_{sk_1}$, and secret keys sk_1, sk_2
Output: Encrypted comparison matrices $[\![\boldsymbol{CR}]\!]_{sk_2}$ and $[\![\boldsymbol{CR'}]\!]_{sk_2}$

1: $\boldsymbol{sr}^{(k)} \leftarrow \text{SHE.Dec}([\![\boldsymbol{sr}^{(k)}]\!]_{sk_1}, sk_1)$
2: $\boldsymbol{sr_matrix} \leftarrow \text{SIMD_decode}(\boldsymbol{sr}^{(k)})$ *// Decode SIMD packed plaintext*
3: $\boldsymbol{CR} \leftarrow \text{check_sign}(\boldsymbol{sr_matrix})$ *// Evaluate mathematical signs*
4: $\boldsymbol{CR'} \leftarrow 1 - \boldsymbol{CR}$ *// Derive complementary indicator matrix*
5: $[\![\boldsymbol{CR}]\!]_{sk_2} \leftarrow \text{SHE.Enc}(\boldsymbol{CR}, sk_2)$
6: $[\![\boldsymbol{CR'}]\!]_{sk_2} \leftarrow \text{SHE.Enc}(\boldsymbol{CR'}, sk_2)$
7: **return** $[\![\boldsymbol{CR}]\!]_{sk_2}, [\![\boldsymbol{CR'}]\!]_{sk_2}$ *// Send them to* CSP

In a plaintext scenario, these binary results straightforwardly dictate the branch assignment for each feature vector. Specifically, a value of 1 routes the instance to the right branch, while a 0 routes it to the left. To execute this routing securely over encrypted data at the CSP side, we formally define the initial comparison matrix $\boldsymbol{CR}_{svc\times d\times n}$ as the routing indicator for the right branch. The DO then algebraically derives the complementary indicator matrix for the left branch, denoted as $\boldsymbol{CR'}_{svc\times d\times n}$, by computing $[1-cr_{k,j,l}]_{svc\times d\times n}$. These two complementary matrices, $\boldsymbol{CR}$ and $\boldsymbol{CR'}$, are subsequently utilized by the CSP to independently aggregate the number of training instances routed to the right and left branches, respectively.

Finally, the DO encrypts each binary element within both $\boldsymbol{CR}$ and $\boldsymbol{CR'}$ using sk_2. The resulting ciphertexts, $[\![\boldsymbol{CR}_{svc\times d\times n}]\!]_{sk_2}$ and $[\![\boldsymbol{CR'}_{svc\times d\times n}]\!]_{sk_2}$, are then transmitted to the CSP.

5.3 The Second Round (T3)

The second interactive round corresponds to Phase T3 in Figure 1, and the overall decision tree construction procedure is formalized in Algorithm 3. This phase is executed iteratively from the root node down to the leaf nodes until the entire decision tree topology is thoroughly generated.

Phase T3.1. At the beginning of this round, the CSP initiates the calculation of the Gini Impurity Index (GII) candidates for the current tree node based on $[\![\boldsymbol{CR}_{svc\times d\times n}]\!]_{sk_2}$, $[\![\boldsymbol{CR'}_{svc\times d\times n}]\!]_{sk_2}$, and $[\![\boldsymbol{LAB}_{n\times c}]\!]_{sk_2}$. Specifically, the CSP performs matrix multiplications between the encrypted comparison results and the encrypted one-hot label matrix to derive the encrypted class accumulation matrices. This homomorphic evaluation is denoted as $[\![\boldsymbol{AR}_{svc\times d\times c}]\!]_{sk_2} = [\![\boldsymbol{CR}_{svc\times d\times n}]\!]_{sk_2} \otimes [\![\boldsymbol{LAB}_{n\times c}]\!]_{sk_2} = [\![\boldsymbol{CR}_{svc\times d\times n} \times \boldsymbol{LAB}_{n\times c}]\!]_{sk_2}$, $[\![\boldsymbol{AR'}_{svc\times d\times c}]\!]_{sk_2} = [\![\boldsymbol{CR'}_{svc\times d\times n}]\!]_{sk_2} \otimes [\![\boldsymbol{LAB}_{n\times c}]\!]_{sk_2} = [\![\boldsymbol{CR'}_{svc\times d\times n} \times \boldsymbol{LAB}_{n\times c}]\!]_{sk_2}$. The matrix $[\![\boldsymbol{AR}_{svc\times d\times c}]\!]_{sk_2}$ (resp. $[\![\boldsymbol{AR'}_{svc\times d\times c}]\!]_{sk_2}$) contains class distribution vectors for various features and split candidates routed to the right (resp. left) branch. For instance, given the j-th feature and k-th split candidate, a class accumulation vector $\boldsymbol{ar} = [5, 8, 7, 0]$ indicates that exactly $5, 8, 7$, and 0 training instances associated with labels $lab_{\mathbf{x}} = 1, 2, 3$, and 4 have been routed to the right branch.

Because the traditional GII calculation necessitates numerous ciphertext divisions, which are computationally prohibitive in SHE, we strategically deploy the Modified Gini Impurity Index (MGII) [11]. The core mathematical reformulation of MGII circumvents division operations when determining the optimal split. Given the j-th feature and k-th split candidate, the MGII at a specific tree node is defined as follows:

$$MGII = \frac{1}{|D|^2}\left((|D_{left}|^2 - \sum_{i=1}^{c}|D_{left,i}|^2) + (|D_{right}|^2 - \sum_{i=1}^{c}|D_{right,i}|^2)\right) \tag{1}$$

where $|D|$ indicates the total number of training instances at the current node, and $|D_{left}|$ (resp. $|D_{right}|$) represents the subset of instances routed to the left

Algorithm 3 Decision Tree Training Algorithm

Participating party: Cloud Service Provider `CSP`
Input: Encrypted one-hot labels $[\![\boldsymbol{LAB}]\!]_{sk_2}$, max tree depth max_depth
Output: Root node of the decision tree $root$

Initialize the $root$ node
$queue \leftarrow \{(root, [\![\boldsymbol{LAB}]\!]_{sk_2}, 0)\}$ *// Initialize Breadth First Search queue*
while $queue \neq \emptyset$ **do**
  $(node, [\![\boldsymbol{LAB}]\!]_{sk_2}, depth) \leftarrow$ queue.pop() *// Extract current node*
  $[\![\boldsymbol{counts}]\!]_{sk_2} \leftarrow \sum_{i=1}^{n} [\![\boldsymbol{LAB}_i]\!]_{sk_2}$ *// Calculate instance statistics*
  if $depth \geq max_depth$ **then**
    $node.value \leftarrow [\![\boldsymbol{counts}]\!]_{sk_2}$ *// Instantiate as leaf node*
    continue
  end if
  $[\![\boldsymbol{CP}]\!]_{sk_2} \leftarrow$ calculate_weighted_gini($[\![\boldsymbol{LAB}]\!]_{sk_2}$)
  Send $[\![\boldsymbol{CP}]\!]_{sk_2}$ to `DO` *// Query optimal split*
  Wait for response from `DO` *// Synchronous interaction*
  if response is termination signal **then** *// Absolute purity reached*
    $node.value \leftarrow [\![\boldsymbol{counts}]\!]_{sk_2}$ *// Instantiate as leaf node*
    continue
  else *// Received split command*
    Obtain $[\![feat_index]\!]_{sk_2}$, $[\![split_index]\!]_{sk_2}$, $[\![\boldsymbol{cr}]\!]_{sk_2}$, $[\![\boldsymbol{cr'}]\!]_{sk_2}$ from `DO`
    $[\![\boldsymbol{LAB_Updt}]\!]_{sk_2} \leftarrow [\![\boldsymbol{LAB}]\!]_{sk_2} \otimes [\![\boldsymbol{cr}]\!]_{sk_2}$ *// Update right branch*
    $[\![\boldsymbol{LAB_Updt'}]\!]_{sk_2} \leftarrow [\![\boldsymbol{LAB}]\!]_{sk_2} \otimes [\![\boldsymbol{cr'}]\!]_{sk_2}$ *// Update left branch*
    Create $node.left$ and $node.right$
    $node.feat_index \leftarrow [\![feat_index]\!]_{sk_2}$
    $node.split_index \leftarrow [\![split_index]\!]_{sk_2}$
    queue.push($node.right, [\![\boldsymbol{LAB_Updt}]\!]_{sk_2}, depth + 1$)
    queue.push($node.left, [\![\boldsymbol{LAB_Updt'}]\!]_{sk_2}, depth + 1$)
  end if
end while
return $root$ *// Send the final tree to* `DO`

(resp. right) branch. Furthermore, $|D_{right,i}|$ (resp. $|D_{left,i}|$) denotes the number of instances bearing the label $lab_{\mathbf{x}} = i$ on the right (resp. left) branch. Since the total instance count $|D|$ at the current node remains strictly constant across all candidate evaluations, minimizing the MGII is mathematically equivalent to minimizing its variable component $cp = ((|D_{left}|^2 - \sum_{i=1}^{c} |D_{left,i}|^2) + (|D_{right}|^2 - \sum_{i=1}^{c} |D_{right,i}|^2))$.

Consequently, the computation of cp requires only ciphertext subtractions and multiplications. The CSP evaluates cp homomorphically for all features and split candidates ($j \in [1, d], k \in [1, svc]$) and subsequently constructs the comprehensive encrypted cp matrix, $[\![\boldsymbol{CP}_{svc\times d}]\!]_{sk_2}$, leveraging $[\![\boldsymbol{AR}]\!]_{sk_2}$, $[\![\boldsymbol{AR'}]\!]_{sk_2}$, and Equation (1).

Phase T3.2. Next, the CSP transmits $[\![\boldsymbol{CP}_{svc\times d}]\!]_{sk_2}$ to the DO. Acting as a cryptographic coprocessor, the DO decrypts the matrix and inspects whether all elements $cp_{k,j}$ evaluate to 0. If true, this indicates a state of absolute purity where all instances at the current node share identical classification labels. The DO then issues a termination signal to the CSP, instructing it to halt further splitting and instantiate a leaf node. Otherwise, the DO identifies the minimum cp value (corresponding to the minimal MGII) within $\boldsymbol{CP}_{svc\times d}$, thereby determining the optimal feature index $j_{best} \in [1, d]$ and the optimal split candidate index $k_{best} \in [1, svc]$.

Using these optimal indices, the DO extracts the corresponding boolean comparison vectors $\boldsymbol{cr}_n^{(k_{best},j_{best})}$ and $\boldsymbol{cr'}_n^{(k_{best},j_{best})}$ from the comparison matrices $\boldsymbol{CR}_{svc\times d\times n}$ and $\boldsymbol{CR'}_{svc\times d\times n}$, respectively. These vectors serve as the definitive routing indicators to partition the training instances into the left and right branches at the current node. Finally, the DO encrypts these optimal parameters using sk_2 and sends $[\![k_{best}]\!]_{sk_2}$, $[\![j_{best}]\!]_{sk_2}$, $[\![\boldsymbol{cr}_n^{(k_{best},j_{best})}]\!]_{sk_2}$, and $[\![\boldsymbol{cr'}_n^{(k_{best},j_{best})}]\!]_{sk_2}$ to the CSP.

Phase T3.3. Upon receiving the optimal indices and routing indicators, the CSP instantiates the current node as an internal node, locally storing $[\![k_{best}]\!]_{sk_2}$ and $[\![j_{best}]\!]_{sk_2}$ along with pointers to its respective child nodes. Before iteratively proceeding to the generation of child nodes, the CSP must logically split the training instances. To achieve this securely, the CSP performs an element-wise multiplication (equivalent to cryptographic masking) between each row of the label matrix $[\![\boldsymbol{LAB}_{n\times c}]\!]_{sk_2}$ and the corresponding elements in $[\![\boldsymbol{cr}_n^{(k_{best},j_{best})}]\!]_{sk_2}$ and $[\![\boldsymbol{cr'}_n^{(k_{best},j_{best})}]\!]_{sk_2}$.

For instance, if the l-th encrypted label is $[\![0, 0, 1, 0]\!]_{sk_2}$, and the corresponding routing indicators are $[\![1]\!]_{sk_2}$ (right) and $[\![0]\!]_{sk_2}$ (left), the homomorphic multiplications yield $[\![0, 0, 1, 0]\!]_{sk_2}$ for the right branch and $[\![0, 0, 0, 0]\!]_{sk_2}$ for the left branch. This mathematical masking assigns the l-th instance exclusively to the right branch, while effectively nullifying its contribution to the accumulation matrix $[\![\boldsymbol{AR}]\!]_{sk_2}$ in subsequent iterative calculations on the left branch. Denoted as $[\![\boldsymbol{LAB_Updt}_{n\times c}]\!]_{sk_2}$ and $[\![\boldsymbol{LAB_Updt'}_{n\times c}]\!]_{sk_2}$, these masked matrices serve as the input datasets for the child nodes. Utilizing these updated label matrices, the CSP and DO iteratively execute Phases T3.1 through T3.3 by dynamically updating the node queue.

Conversely, if the CSP receives a termination signal from the DO or if the tree depth reaches a predefined threshold, the CSP finalizes the current node as a leaf. Because evaluating plaintext magnitude conditions over ciphertexts is impossible, the CSP aggregates the elements in $[\![\boldsymbol{LAB}_{n\times c}]\!]_{sk_2}$ column-wise to compute the class distribution vector $[\![\boldsymbol{num}_c]\!]_{sk_2}$. For example, an encrypted summation yielding $[\![3,5,4,0]\!]_{sk_2}$ securely encapsulates the distribution without leaking it. Ultimately, the DO will be responsible for decrypting $[\![\boldsymbol{num}_c]\!]_{sk_2}$ during model retrieval to determine the predominant classification label (e.g., class 2) for that leaf.

Upon fully expanding the decision tree topology, the CSP transmits the entirely encrypted tree structure, comprising all internal and leaf nodes, back to the DO.

5.4 The Third Round (T4)

The third round corresponds to the Phase T4 in Figure 1. The DO receives and decrypts all the encrypted internal and leaf tree nodes.

- For a internal tree node, after decryption, the DO obtains the indices of the selected feature and split value candidate (i.e., k_{best} and j_{best}). Based on these indices, the DO can directly retrieve the corresponding feature and split value for each internal node.
- For a leaf tree node, after decryption, the DO obtains the class distribution vector $\boldsymbol{num}_c$ (e.g., $[3,5,4,0]$), which represents the number of training instances associated with each class. The predicted label is determined by selecting the class with the maximum count (e.g., class 2 corresponding to 5).

At this point, the DO has recovered the complete decision tree model, including the feature and split value at each internal node and the classification result at each leaf node.

6 Experiment Results and Analysis

We evaluated our scheme on two virtual machines running Ubuntu 20.04. The DO was deployed on a VM with an 8-core Intel Xeon Gold 6132 CPU (2.6 GHz) and 16 GB RAM, while the CSP ran on a VM with a 24-core Intel Xeon Silver 4214 CPU (2.2 GHz) and 64 GB RAM. Compared with the hardware configurations reported in related work, our setup is relatively modest, yet our scheme achieves better time performance.

To ensure a fair and rigorous comparison with closely related work (i.e., Akavia et al. [5] and Shin et al. [24]), we selected four benchmark datasets from the standard UCI repository [17], namely Iris [2], Wine [3], Breast Cancer [1], and Sepsis [14]. The dataset statistics are summarized in Table 2. For each dataset, 70% of the instances were used for training and the remaining 30% for testing.

Table 2: Our Training Dataset Statistics

Dataset	Features No.	Classes No.	Training ins.	Testing ins.
Iris [2]	4	3	105	45
Wine [3]	13	3	124	54
Breast Cancer [1]	30	2	398	171
Sepsis [14]	3	2	77239	33102

The choice of cryptographic parameters affects both security and efficiency. In our implementation, sk_1 was set to $(k_0, k_1, k_2) = (1537, 600, 768)$, and sk_2 was set to $(k_0, k_1, k_2) = (2080, 30, 90)$. A detailed security analysis of these parameters is provided in Section 6.6.

To enable efficient integer computation, all feature values were mapped into a unified discrete space. Continuous features were quantized and normalized to the range $[-2^{14}, 2^{14})$, while discrete features were directly represented within this range. Each encoded value was padded with one zero bit to prevent overflow. With a plaintext capacity of $k_1 = 600$ bits, each feature occupied 15 bits plus one padding bit, allowing $\lfloor 600/16 \rfloor = 37$ feature values to be packed into a single ciphertext for SIMD processing.

6.1 Model Accuracy and Comparison

As shown in Figure 2, our scheme achieves up to 98% accuracy on multiple datasets when the tree depth was set to 5 or 6. Compared with closely related work (i.e., Akavia et al. [5] and Shin et al. [24]), which reported results only on the Iris, Wine, and Breast Cancer datasets, our method consistently achieves higher accuracy for deeper trees.

This performance difference is mainly due to the underlying cryptographic primitives. Both Akavia et al. [5] and Shin et al. [24] rely on the CKKS scheme, which performs approximate homomorphic computation. As a result, polynomial approximations are used during ciphertext comparison, introducing approximation errors. In contrast, our scheme is based on exact integer arithmetic in SHE. By delegating comparison operations to the `DO` through our interactive protocol, we avoid approximate comparisons and eliminate the associated accuracy loss in this phase.

In general, our scheme achieves accuracy comparable to the standard plaintext CART baseline, and in some cases, even outperforms it. This phenomenon can be explained by the regularization effect introduced by data quantization and discretization. While standard CART searches for split points over continuous values with high precision, it may overfit by capturing noise in the data. In contrast, our scheme restricts split points to a predefined discrete set, which acts as an implicit regularizer. Moreover, computing MGII in the integer domain further improves robustness by reducing sensitivity to minor statistical variations. As a

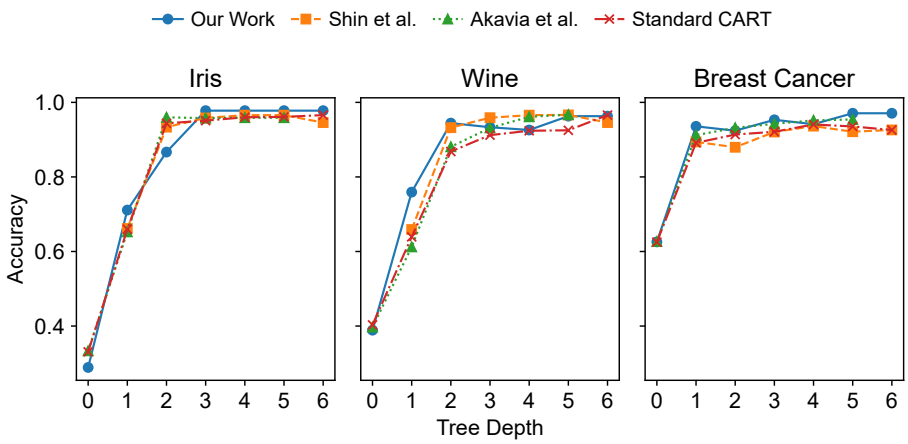


Fig. 2: Accuracy comparison between our scheme and related works

result, the model focuses more on dominant data patterns, leading to improved generalization on unseen data.

6.2 Communication Overhead of Our Scheme

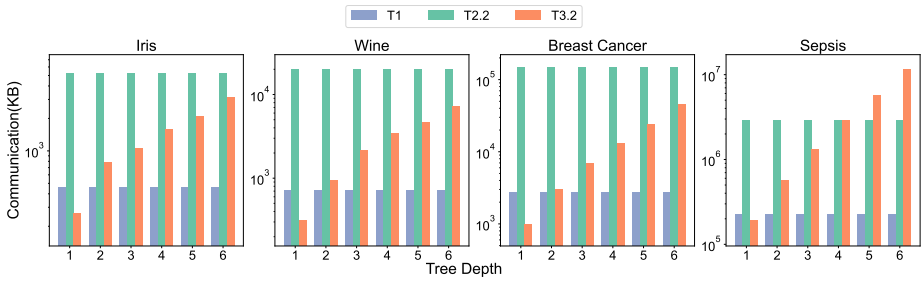
We evaluated the communication overhead of our protocol, with results shown in Figure 3.

The communication from the DO to the CSP (Figure 3(a)) consists of three components, corresponding to the three interactions shown in Figure 1.

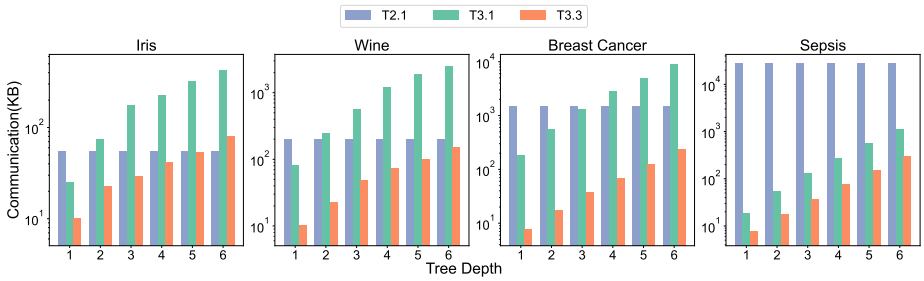
- *Phase T1.* The DO sends encrypted initialization data, including public keys, the encrypted dataset, one-hot labels, and split candidates.
- *Phase T2.2.* The DO returns encrypted comparison results derived from feature–candidate subtractions.
- *Phase T3.2.* The DO transmits encrypted indices of the selected feature and split, along with the corresponding comparison result vector. This component grows logarithmically with the tree depth.

The communication from the CSP to the DO (Figure 3(b)) also consists of three components.

- *Phase T2.1.* The CSP sends encrypted subtraction results between feature values and split candidates.
- *Phase T3.1.* The CSP transmits encrypted data required for MGII computation.
- *Phase T3.3.* The CSP sends encrypted internal and leaf nodes of the tree. The latter two components grow logarithmically with the tree depth.



(a) Communication overhead details (from DO to CSP)



(b) Communication overhead details (from CSP to DO)

Fig. 3: The details of communication overhead

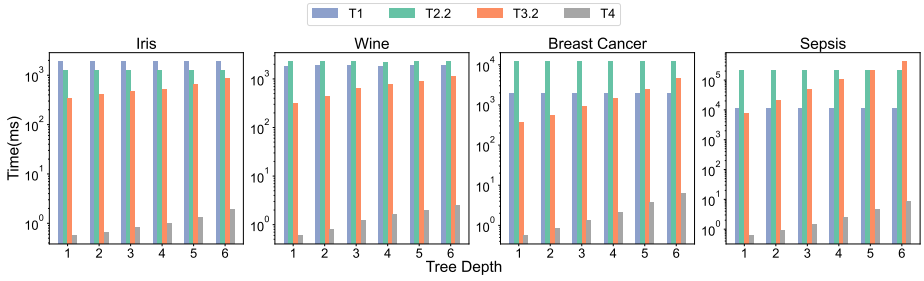
In summary, all transmitted data remain encrypted, and the communication overhead grows either linearly or logarithmically with the tree depth. As shown in Section 6.3, the total execution time is dominated by computation rather than communication under typical network conditions.

6.3 Runtime Performance and Comparison

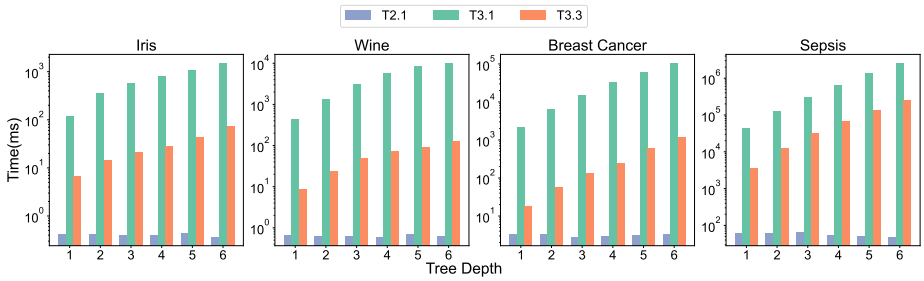
We evaluated both the computation time of the DO and CSP, as well as the overall execution time of our protocol, which includes both computation and communication. All results were averaged over 10 independent runs.

The computation time of the DO (Figure 4(a)) is divided into four phases.

- *Phase T1.* Initialization, including key generation, data normalization, and SIMD-based encryption.
- *Phase T2.2.* Decryption of subtraction results, sign evaluation, and encrypting the comparison outputs again. This phase incurs a constant-time cost for a fixed dataset and candidate set.
- *Phase T3.2.* Decryption of the MGII matrix, selection of the optimal feature-split pair, and encryption of the selected indices.
- *Phase T4.* Final decryption to obtain the plaintext tree structure.



(a) Computation time overheads of DO



(b) Computation time overheads of CSP

Fig. 4: The details of computation time overheads

As shown in Figure 4(a), the computation time of the DO scales with the dataset size. For small datasets such as Iris and Wine, each phase took around 10^3 ms, resulting in a total execution time of a few seconds. For larger datasets such as Breast Cancer and Sepsis, the overhead increases accordingly. Among all phases, T2.2 dominates the computation time, as it involves decryption, sign evaluation, and re-encryption over the large comparison matrix $\boldsymbol{CR}$, reaching approximately 10^4 to 10^5 ms.

The computation time of the CSP (Figure 4(b)) is divided into three phases.

- *Phase T2.1.* Parallel ciphertext subtractions between feature values and split candidates.
- *Phase T3.1.* Computation of encrypted MGII values, involving a large number of homomorphic multiplications.
- *Phase T3.3.* Tree construction and data routing, which also requires substantial homomorphic multiplications.

As shown in Figure 4(b), the computation time of the CSP increases exponentially with the tree depth, due to the doubling number of nodes at each level. Phase T2.1 remains efficient and nearly constant, while Phase T3.1 dominates the total cost because of intensive homomorphic multiplications. For small datasets (e.g., Iris and Wine), the execution time ranged from 10^2 to 10^4 ms.

For larger datasets and deeper trees (e.g., Sepsis with depth 6), the total time reached around 10^6 ms.

Since our protocol involves multiple interaction rounds, communication latency may affect the overall performance. To evaluate this impact, we simulated two network settings, namely a high-speed LAN (1 Gbps, 0.2 ms RTT) and a standard Internet environment (100 Mbps, 10 ms RTT). The total execution time, including both computation and communication, is reported in Figure 5. The results show that our scheme consistently achieves lower total execution time than closely related work (i.e., Akavia et al. [5] and Shin et al. [24]), even under standard network conditions. This indicates that the computational efficiency of our approach effectively offsets the additional communication overhead.

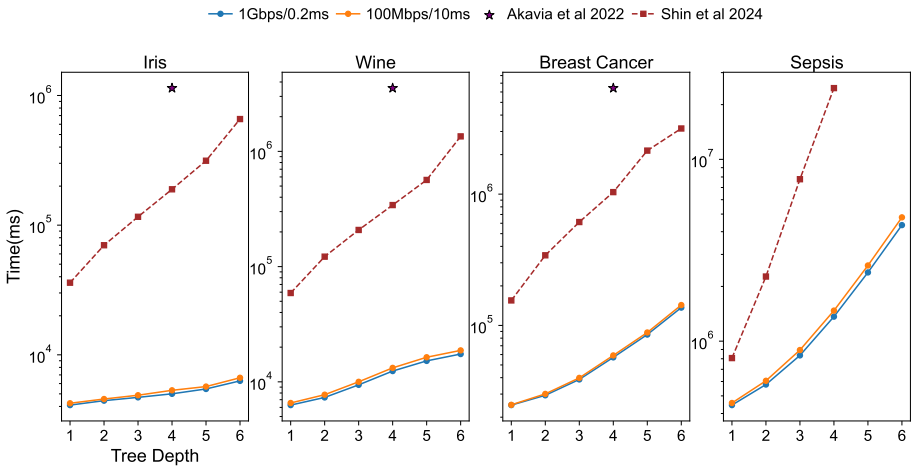


Fig. 5: Comparison of total execution time (including computation and communication)

6.4 Computational Complexity and Efficiency Analysis

To explain the observed performance differences, we analyzed the computational complexity and the efficiency of the underlying cryptographic operations in our scheme and related work. The notations used in this analysis are summarized below.

- n, d, c: The number of training instances, features, and classes, respectively.
- svc, m: The number of split value candidates and total tree nodes.
- S: The SIMD packing factor (denoted as $S_{Akavia}, S_{Shin}, S_{Ours}$).
- T_{poly}: The cost of evaluating high-degree approximation polynomials (e.g., degree 32 or 64) used in CKKS-based schemes.

- $T_{enc}, T_{mult}, T_{dec}$: The cost of operations in CKKS-based schemes.
- $T'_{enc}, T'_{mult}, T'_{dec}$: The cost of operations in our SHE-based scheme.

Table 3 summarizes the asymptotic time complexity of different schemes. Our scheme has a similar asymptotic form to related work. The key differences lie in the SIMD packing factor (S) and the cost of a single ciphertext operation (T_{op}). These two factors jointly determine the amortized cost per data item (T_{op}/S), which ultimately dominates the overall runtime. Although our scheme uses a smaller packing factor, it benefits from significantly faster ciphertext operations (i.e., $T'_{mult} \ll T_{mult}$, $T'_{mult} \ll T_{poly}$, $T'_{enc} \ll T_{enc}$, and $T'_{dec} \ll T_{dec}$). As a result, our scheme achieves a substantially lower amortized cost, leading to better practical performance.

Table 3: Comparison of time complexity between our scheme and related work

Schemes	**CSP Time Complexity**	**DO Time Complexity**
Akavia et al. [5]	$\mathcal{O}\left(\frac{n}{S_{Akavia}} \cdot m \cdot svc \cdot d \cdot c\right) \cdot T_{poly}$	$\mathcal{O}\left(\frac{n}{S_{Akavia}}(d+c) + m \cdot svc \cdot d \cdot c\right) \cdot (T_{dec} + T_{enc})$
Shin et al. [24]	$\mathcal{O}\left(\frac{n}{S_{Shin}} \cdot svc \cdot d \cdot c\right) \cdot T_{mult}$ $+O(m \cdot svc \cdot d) \cdot T_{poly}$	$\mathcal{O}\left(\frac{n}{S_{Shin}} \cdot (svc \cdot d + c)\right) \cdot T_{enc}$
Our Scheme	$\mathcal{O}\left(\frac{n}{S_{Ours}} \cdot m \cdot svc \cdot d \cdot c\right) \cdot T'_{mult}$	$\mathcal{O}\left(\frac{n}{S_{Ours}}(d + c + svc \cdot d + m)\right) \cdot (T'_{dec} + T'_{enc})$

We conducted micro-benchmarks on a workstation with an Intel i5-13400F CPU and 32 GB RAM. The results are shown in Table 4. CKKS-based schemes incur millisecond-level latency per operation due to polynomial arithmetic, and therefore rely on large packing factors (e.g., as reported in [5, 24], $S_{Akavia} = 8192$ and $S_{Shin} = 16384$) to amortize the cost. In contrast, our SHE-based scheme performs lightweight modular arithmetic, achieving microsecond-level latency. This allows efficient execution even with a smaller packing factor (e.g., $S_{Ours} = 37$).

Table 4: Overheads of atomic cryptographic operations

Operations	**Our Scheme**		**Akavia et al. [5]**		**Shin et al. [24]**	
	Single OP	**Amortized**	**Single OP**	**Amortized**	**Single OP**	**Amortized**
Encryption	5.72 μs	**0.1550** μs	7.24 ms	0.8838 μs	18.16 ms	1.1084 μs
Decryption	1.89 μs	**0.0512** μs	2.24 ms	0.2734 μs	6.05 ms	0.3695 μs
Multiplication	0.58 μs	**0.0156** μs	5.28 ms	0.6451 μs	16.83 ms	1.0274 μs

These results confirm that the amortized cost per data item (T_{op}/S) plays a key role in overall efficiency. Even after amortization, our scheme maintains a clear advantage. For example, the amortized multiplication cost is $0.0156\,\mu s$, compared to $0.6451\,\mu s$ in [5] and $1.0274\,\mu s$ in [24], corresponding to speedups of approximately $40\times$ and $65\times$, respectively. Similar trends are observed for encryption and decryption.

The efficiency gap is more evident in comparison operations. CKKS-based schemes require evaluating high-degree approximation polynomials, which introduces substantial computational overhead. For example, [5] reported that a degree-32 soft-step function requires 0.594 seconds per operation, or $72.5\,\mu s$ after amortization. Similarly, [24] reported an ApproxSign cost of 16,995 ms, corresponding to approximately $1{,}037\,\mu s$ per item after packing.

In contrast, our scheme replaces these polynomial evaluations with an interactive protocol involving decryption and re-encryption. This reduces the cost to approximately $0.05\,\mu s$ and $0.15\,\mu s$ per item, respectively. Although this design introduces interaction, the associated cost remains small compared to polynomial-based approaches. In addition, our scheme avoids the need for bootstrapping, further improving efficiency.

Overall, based on the complexity analysis, our design leads to significantly improved practical performance.

6.5 Scalability Analysis on a Large-Scale Dataset

In previous experiments, we used standard UCI datasets for fair comparison with related work. However, these datasets are relatively small. To evaluate scalability of our scheme under larger workloads, we conducted experiments on the Forest Covertype dataset (binary classification), with up to 75,000 instances and 54 features. Due to the increased computational demand, these experiments were conducted on a workstation with 32-core Intel Xeon Platinum 8352V CPU and 120 GB RAM. We analyzed scalability by varying the training size (n), feature dimension (d), and number of split candidates (svc). Results are shown in Figure 6.

We first varied the training size n from 15,000 to 75,000 with $d = 54$ and $svc = 5$. As shown in Figures 6(a) and (d), both runtime and memory usage increase approximately linearly with n. When n increases by $5\times$, the runtime increases from 4,154 seconds to 20,643 seconds (about $4.96\times$), consistent with our complexity analysis. The peak memory usage of the `CSP` reaches 89.36 GB at $n = 75{,}000$, while the `DO` requires only 3.63 GB due to a pipelined design that reduces client-side memory overhead.

Then, we varied the feature dimension $d \in \{10, 20, 30, 40, 54\}$ with $n = 10{,}000$ and $svc = 5$. As shown in Figures 6(b) and (e), both runtime and memory scale linearly with d. Even at $d = 54$, the runtime remains around 2,943 seconds.

Finally, we varied the number of split candidates $svc \in \{3, 5, 9, 17, 33\}$ with $n = 10{,}000$ and $d = 54$. As shown in Figures 6(c) and (f), both runtime and memory usage increase linearly with svc. We also observed a trade-off between efficiency and accuracy. Increasing svc from 3 to 33 raises the runtime from 1,725

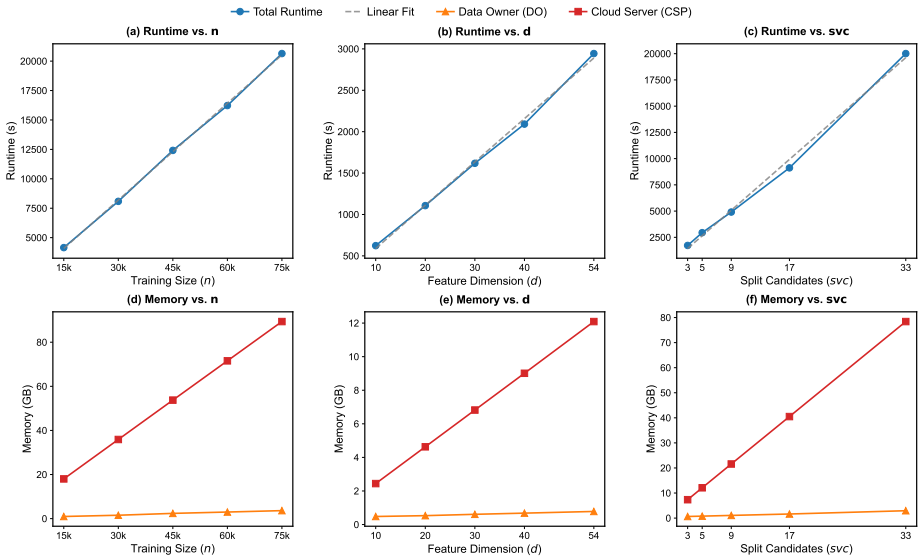


Fig. 6: Scalability analysis on the Forest Covertype dataset. Top: runtime; bottom: peak memory usage. From left to right: varying n, d, and svc.

seconds to 20,012 seconds, while improving accuracy from 62.20% to 73.52%. Due to the efficiency of our integer-based operations, larger svc values can be used to improve accuracy with moderate overhead.

Overall, the results demonstrate that our scheme scales approximately linearly with respect to n, d, and svc, which indicates that our scheme can effectively support privacy-preserving training on large datasets in practice.

6.6 Security Analysis

As shown in [27], the underlying SHE scheme employed in our protocol achieves IND-CPA (indistinguishability under chosen-plaintext attack) security under the $(\mathcal{R}, p)$-decision assumption. This security guarantee holds provided that the bit-lengths of the parameters $\mathcal{R}$ and p (denoted by k_2 and k_0, respectively) satisfy appropriate cryptographic requirements. Specifically, $\mathcal{R}$ is a random integer of bit-length k_2, and it is generally required that $k_2 \geq 90$ to prevent exhaustive search attacks. The parameter p is a large prime, and together with another prime q, forms the public modulus $N = pq$. Since N is public, recovering p reduces to the integer factorization problem. To ensure security, the modulus N should be at least 1024 bits, implying that $k_0 \geq 512$.

In our protocol, we instantiate two independent SHE key pairs with parameters $k_2 \in \{90, 768\}$ and $k_0 \in \{1537, 2080\}$. All parameter choices meet or exceed the above security requirements, thereby ensuring IND-CPA security. Based on

this foundation, we analyzed the privacy guarantees of our protocol under the honest-but-curious threat model (Section 4.2).

- **Training Dataset Privacy**: The training dataset is encrypted by the DO before outsourcing. Specifically, feature data and split candidates are encrypted under (pk_1, sk_1), while one-hot encoded labels are encrypted under (pk_2, sk_2). Since the CSP does not possess the corresponding secret keys, it cannot decrypt the ciphertexts or infer any underlying plaintext information, ensuring data confidentiality.
- **DT Model Privacy**: During the entire training process, all model-related information, including selected features, split candidates, leaf labels, and intermediate results, remains encrypted under (pk_2, sk_2). The CSP performs only homomorphic operations on ciphertexts. Due to the IND-CPA security of the SHE scheme, it learns no information about intermediate computations, the final tree structure, or the secret keys, thereby preserving model privacy.

7 Conclusion

In this paper, we propose an efficient and privacy-preserving outsourced decision tree training framework based on an integer-based SHE scheme. Our protocol enables the DO to securely delegate training tasks to a CSP while preserving both data and model confidentiality. By proposing an integer-based modified GII suitable for SHE and integrating SIMD batching, we effectively overcome the inefficiencies of polynomial-based approaches.

Extensive experiments demonstrate that our scheme achieves comparable accuracy overall, with a clear advantage for deeper trees, while maintaining a lightweight memory footprint for the DO and significantly reducing computational overhead. Moreover, even when accounting for communication latency under standard network conditions, the overall execution time remains substantially lower than that of related works. These results indicate that our lightweight interactive design eliminates the need for costly bootstrapping and polynomial approximations, making the proposed scheme practical and scalable for real-world privacy-preserving machine learning applications. This work provides a promising direction for efficient and accurate privacy-preserving outsourced learning under practical constraints.

Acknowledgments. This work is supported by the Fundamental Research Funds for the Central Universi-ties (Program No. 2662022XXYJ004).

References

1. Breast cancer (Accessed: Mar 2026 [Online]), available: http://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic

2. Iris (Accessed: Mar 2026 [Online]), available: http://archive.ics.uci.edu/dataset/53/iris
3. Wine (Accessed: Mar 2026 [Online]), available: https://archive.ics.uci.edu/dataset/109/wine
4. Adams, S., Choudhary, C., De Cock, M., Dowsley, R., Melanson, D., Nascimento, A., Railsback, D., Shen, J.: Privacy-preserving training of tree ensembles over continuous data. Proceedings on Privacy Enhancing Technologies **2022**, 205–226 (04 2022). https://doi.org/10.2478/popets-2022-0042
5. Akavia, A., Leibovich, M., Resheff, Y.S., Ron, R., Shahar, M., Vald, M.: Privacy-preserving decision trees training and prediction. ACM Trans. Priv. Secur. **25**(3) (may 2022). https://doi.org/10.1145/3517197
6. Akhavan Mahdavi, R., Ni, H., Linkov, D., Kerschbaum, F.: Level up: Private non-interactive decision tree evaluation using levelled homomorphic encryption. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. p. 2945–2958. CCS '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3623095
7. Azogagh, S., Delfour, V., Gambs, S., Killijian, M.O.: Probonite: Private one-branch-only non-interactive decision tree evaluation. In: Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography. p. 23–33. WAHC'22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3560827.3563377
8. Bhardwaj, D., Saravanan, S., Chandran, N., Gupta, D.: Securely training decision trees efficiently. In: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security. pp. 4673–4687 (2024)
9. Brakerski, Z., Gentry, C., Halevi, S.: Packed ciphertexts in lwe-based homomorphic encryption. In: Kurosawa, K., Hanaoka, G. (eds.) Public-Key Cryptography – PKC 2013. pp. 1–13. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
10. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
11. Bădulescu, L.A.: Experiments for a better gini index splitting criterion for data mining decision trees algorithms. In: 2020 24th International Conference on System Theory, Control and Computing (ICSTCC). pp. 208–212 (2020). https://doi.org/10.1109/ICSTCC50638.2020.9259691
12. Cascudo, I., Gundersen, J.S.: A secret-sharing based MPC protocol for boolean circuits with good amortized complexity. In: Pass, R., Pietrzak, K. (eds.) Theory of Cryptography. pp. 652–682. Springer International Publishing, Cham (2020)
13. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 409–437. Springer International Publishing, Cham (2017)
14. Chicco, D., Jurman, G.: Sepsis Survival Minimal Clinical Records. UCI Machine Learning Repository (2020), DOI: https://doi.org/10.24432/C53C8N
15. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
16. Cong, K., Das, D., Park, J., Pereira, H.V.: Sortinghat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 563–577. CCS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3548606.3560702
17. Dua, D., Graf, C.: Uci machine learning repository (2017), http://archive.ics.uci.edu/

18. Frery, J., Stoian, A., Bredehoft, R., Montero, L., Kherfallah, C., Chevallier-Mames, B., Meyre, A.: Privacy-preserving tree-based inference with TFHE. In: Bouzefrane, S., Banerjee, S., Mourlin, F., Boumerdassi, S., Renault, É. (eds.) Mobile, Secure, and Programmable Networking. pp. 139–156. Springer Nature Switzerland, Cham (2024)
19. Lin, G., Han, W., Ruan, W., Zhou, R., Song, L., Li, B., Shao, Y.: Ents: an efficient three-party training framework for decision trees by communication optimization. In: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security. pp. 4376–4390 (2024)
20. Liu, L., Chen, R., Liu, X., Su, J., Qiao, L.: Towards practical privacy-preserving decision tree training and evaluation in the cloud. IEEE Transactions on Information Forensics and Security **15**, 2914–2929 (2020). https://doi.org/10.1109/TIFS.2020.2980192
21. Loh, W.Y.: Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **1**, 14 – 23 (01 2011). https://doi.org/10.1002/widm.8
22. jie Lu, W., Huang, Z., Zhang, Q., Wang, Y., Hong, C.: Squirrel: A scalable secure Two-Party computation framework for training gradient boosting decision tree. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 6435–6451. USENIX Association, Anaheim, CA (Aug 2023)
23. Mahdikhani, H., Lu, R., Zheng, Y., Shao, J., Ghorbani, A.A.: Achieving o($\log^3$n) communication-efficient privacy-preserving range query in fog-based IoT. IEEE Internet of Things Journal **7**(6), 5220–5232 (2020). https://doi.org/10.1109/JIOT.2020.2977253
24. Shin, H., Choi, J., Lee, D., Kim, K., Lee, Y.: Fully homomorphic training and inference on binary decision tree and random forest. In: García-Alfaro, J., Kozik, R., Choras, M., Katsikas, S.K. (eds.) Computer Security - ESORICS 2024 - 29th European Symposium on Research in Computer Security, Bydgoszcz, Poland, September 16-20, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14984, pp. 217–237. Springer (2024). https://doi.org/10.1007/978-3-031-70896-1_11
25. Tueno, A., Boev, Y., Kerschbaum, F.: Non-interactive private decision tree evaluation. In: Singhal, A., Vaidya, J. (eds.) Data and Applications Security and Privacy XXXIV. pp. 174–194. Springer International Publishing, Cham (2020)
26. Zhao, J., Zhu, H., Wang, F., Lu, R., Liu, Z., Li, H.: Pvd-fl: A privacy-preserving and verifiable decentralized federated learning framework. IEEE Transactions on Information Forensics and Security **17**, 2059–2073 (2022). https://doi.org/10.1109/TIFS.2022.3176191
27. Zheng, Y., Lu, R., Guan, Y., Shao, J., Zhu, H.: Efficient and privacy-preserving similarity range query over encrypted time series data. IEEE Transactions on Dependable and Secure Computing **19**(4), 2501–2516 (2022). https://doi.org/10.1109/TDSC.2021.3061611
28. Zheng, Y., Duan, H., Wang, C., Wang, R., Nepal, S.: Securely and efficiently outsourcing decision tree inference. IEEE Transactions on Dependable and Secure Computing **19**(3), 1841–1855 (2022). https://doi.org/10.1109/TDSC.2020.3040012