Privacy-Preserving Decision Tree Inference in a Dual-Cloud Outsourcing Model

Xiaoke Zhou[1], Qianxing Li[1], Chuanyun Dai[1], Bingwei Wang[1], and Yuheng Xia[1]
[1]School of Computer Science and Technology, Xidian University, Xi'an, 710071, Shaanxi Province, China

Privacy-preserving Decision Tree Evaluation (PDTE) enables clients to classify their private data using a decision tree classification model hosted on a server, without revealing the data or classification results. This provides a feasible and secure alternative to traditional decision tree evaluation methods. However, existing solutions often rely on complex operations such as homomorphic encryption (HE) and garbled circuits (GC), which lead to significant computational and communication overhead when performing privacy-preserving inference tasks on large-scale decision tree models. This results in difficulties in balancing data security, inference accuracy, and computational efficiency. To address this issue, this paper proposes a secure and efficient privacy-preserving decision tree outsourcing inference scheme based on secret sharing within semi-honest dual-cloud outsourcing model (SS-PDI). Our scheme is roughly divided into four phases: in the preparation phase, the privacy-preserving decision tree is initialized by hiding the access patterns of the nodes through ciphertext-oriented computational improvements based on the traditional decision tree. In the feature selection phase, the feature selection protocol is improved by introducing the oblivious transfer technique, which reduces the number of selected features and improves the operational efficiency. In the comparison phase, a lightweight Boolean circuit design is used to effectively reduce the overhead of the secure comparison operation. In the evaluation phase, the dual cloud servers in the system return the inference results to the client in the form of secret sharing. After experimental validation, our scheme achieves 74.8% and 40.8% improvement in runtime over complex datasets in LAN and in WAN compared to previous schemes of the same type. There is also a large improvement in the runtime on public datasets compared to previous schemes.

***Index Terms*—Decision Tree, Security Outsourcing, Privacy Protection, Arithmetic Secret Sharing, Boolean Secret Sharing.**

I. Introduction

With the rapid development of cloud computing, an increasing number of enterprises are leveraging cloud infrastructure for data storage and online analysis [1]. In this emerging service paradigm, model providers can deploy pre-trained models on cloud servers, offering inference and prediction services to clients,thereby delivering significant benefits and convenience. Among the various machine learning models, the decision tree model has gained widespread popularity due to its ease of use, versatility in addressing both classification and regression tasks, and its ability to handle diverse data types. It has been successfully applied in numerous practical domains, such as medical diagnosis [2], credit risk assessment[3], e-commerce recommendations, and spam filtering.

However, outsourcing decision tree evaluation [4], [5] to the cloud raises significant privacy concerns regarding both the model and the input data. It is crucial that the model remains confidential and that the model provider does not have access to the client's input data. This necessitates the use of Privacy-preserving Decision Tree Evaluation (PDTE) techniques to prevent any leakage of information related to the client's input or the decision tree itself. If the model parameters are compromised, an attacker could potentially reconstruct similar models, thus infringing on the provider's intellectual property.Furthermore, if user privacy or prediction results are exposed, it could not only adversely impact the user's personal life but also erode trust in the provider, ultimately damaging the user-provider relationship [6], [7]. In conclusion, while outsourced decision tree computation offers convenience, it simultaneously presents serious security and privacy challenges.

In response to the aforementioned challenges, this paper proposes SS-PDI, a secure and efficient privacy-preserving decision tree outsourcing inference scheme based on secret sharing within a dual-cloud outsourcing model. SS-PDI effectively integrates advanced cryptographic techniques with PDTE, offering an innovative solution that ensures the security of both the model and client data while satisfying practical performance requirements. By utilizing the strengths of secret sharing and a dual-cloud architecture, SS-PDI ensures that sensitive information remains protected throughout the inference process. Additionally, it strikes an effective balance between security and computational efficiency, making it a promising solution for real-world applications where both privacy and performance are critical.

Our contributions can be summarized as follows:

- We enhanced the data structure of the decision tree within the system model by adapting it for ciphertext computation, thereby obscuring the number of comparisons during the inference stage. Building upon this improvement, we also optimized the feature selection module using Oblivious Transfer (OT) technology. This modification effectively reduces the number of features selected during the inference process, thereby further enhancing the system's efficiency.
- We design a secure comparison circuit based on Boolean sharing to reduce the computational overhead associated with secure comparisons. This approach leverages the efficiency of Boolean circuits, enabling the secure comparison of data without revealing sensitive information.

Manuscript received December 10, 2024; revised October 11, 2025. Corresponding author: Xiaoke Zhou (email: 20049200456@stu.xidian.edu.cn).

By using Boolean sharing, we ensure that the comparison process remains secure while minimizing the computational cost, thereby enhancing the overall efficiency of the privacy-preserving decision tree inference system.
- We redesign the protocol based on a semi-honest model, which significantly improves both computational and communication performance when applied to large decision trees. This new design not only enhances efficiency but also maintains a high level of security, outperforming existing solutions in terms of scalability and resource utilization.

The rest of this paper is organized as follows: In Section II, we review the related work. In Section III, we present preliminaries of this paper, decision tree and cryptographic primitives. The description of the system model and security goals is given in Section IV. In Section V, we describe the model preparing for secure inferences. In Section **??**, we give the security and performance analysis of the scheme proposed. In Section VII, we present the experimental results. Finally, the paper is concluded in Section VIII.

II. Related Work

Early research in PDTE focused on utilizing Homomorphic Encryption (HE) and Garbled Circuits (GC) to ensure privacy. Bost et al. [8] represent decision trees as higher-order polynomials, which were encrypted using the computationally intensive Leveled Full Homomorphic Encryption (LFE) scheme. Wu et al. [9] proposed a more cost-effective protocol, relying exclusively on OT and Additive Homomorphic Encryption (AHE). However, this approach necessitated the conversion of the decision tree into a full binary structure to obscure its topology, resulting in substantial communication and computational overhead. To mitigate these issues, Tai et al. [10] introduced a path cost mechanism using AHE, which improved performance for sparse trees, though it still incurred a linear cost. Zheng et al. [11] further advanced the field by proposing a dual-cloud outsourcing scheme based on additive secret sharing, which replaced Tai's pure AHE approach and outsourced the computations to cloud servers. In their subsequent work [12], they optimized the feature selection and security comparison modules , reducing both the communication complexity and the interaction time with cloud servers.Guo et al. [13] Significant speedup by running improved nonlinear functions on GPUs. Song et al. [14] propose L-SecNet, which combines additive secret sharing and multiplicative secret sharing to design a lightweight secure comparison protocol, reducing latency caused by comparison operations in non-linear layers. However, this dual-cloud solution is limited to complete binary trees and uses virtual nodes to handle sparse trees, leading to exponentially increasing computational and communication overhead as the tree depth grows.

Several recent studies have sought to reduce the computational complexity of decision tree protocols to sublinear levels. Joye and Salehi [15] achieve this by reducing the number of secure comparisons to d utilizing the DGK protocol instantiated with AHE [16] for secure comparisons. For a tree with l layers, the communication cost is $O(2^{d-1})$, making this protocol sublinear in computation but still incurring significant communication overhead. Tueno et al. [17] represent the decision tree as an array and employed Oblivious Array Indexing (OAI) to securely select the desired tree node. OAI can be instantiated using GC, OT, or Oblivious RAM (ORAM). The first two OAI instances achieve sublinear complexity only for feature providers, as they require OT protocols to select the target node from 2^d possible nodes. However, Using ORAM leads to a communication cost of $O(d^4)$ and requires d^2 rounds, resulting in higher communication overhead. Ma et al. [18] introduced a lightweight protocol in which the tree holder encrypts the decision tree and sends it to the feature provider. At each tree layer, an OT protocol and secure comparison are performed, and the feature provider searches for the next node within the locally encrypted tree. This design significantly reduces communication overhead by moving the most costly selection operation to local computation. However, the feature provider may gain some information from the memory access patterns across multiple evaluations. To preserve security, the decision tree must be re-randomized and re-sent to the feature provider for each evaluation, resulting in linear communication and computational complexity.

To further mitigate the communication and computational overhead in PDTE, some researches have focused on participant assumptions, particularly in the context of semi-honest adversaries. Early works [19], [20], [21], [9] employed HE for decision tree evaluation, but the associated high computational and communication costs rendered these approaches impractical. Tai et al. [10] introduce the concept of path cost to reduce communication overhead without traversing the entire tree. Later works [22], [23], [24] further enhanced computational efficiency, albeit at the cost of some security. Ji et al. [25] proposed an efficient PDTE approach using Function Secret Sharing (FSS), based on Distributed Point Functions (DPF) and Distributed Comparison Functions (DCF). This approach incorporates OT and Conditional Oblivious Transfer (COT) protocols, achieving optimal communication complexity. However, while the protocol ensures security, it introduces significant computational overhead. Upon this research, Fu et al. [26] build prefix parity DICF. This approach significantly reduces both computational and communication overhead, even in the presence of a malicious participant.

The above discussion on existing privacy-preserving decision tree protocols is summarized in Table I. The m in the table represents the number of nodes in the decision tree, n represents the length of the incoming feature vector, d represents the depth of the decision tree, and l represents the number of bits in the binary representation of the parameter.

III. Preliminaries

A. Decision Tree

Decision tree is a commonly used machine learning algorithm, primarily applied to classification and regression tasks. Its goal is to build a tree-like structure by recursively splitting the dataset into smaller, more distinguishable subsets, composed of decision nodes and leaf nodes. Each decision

TABLE I
COMPARISON OF EXISTING PRIVACY-PRESERVING DECISION TREE SOLUTIONS

Protocols	Comparisons	Communication Overhead	Rounds	Encryption Primitives
Bost[8]	$\lceil m/2 \rceil$	$O(n+m)$	≥ 6	Leveled-FHE
Wu[9]	2^d	$O(2^d+(n+m)l)$	6	AHE, OT
Tai[10]	$\lceil m/2 \rceil$	$O((n+m)l)$	4	AHE
Zheng[11]	2^d	$O((n \cdot m)l)$	$l+d$	SS
Zheng[12]	2^d	$O((n+m)l)$	$\log_2 l + d$	SS
Joye[15]	d	$O(d(l+n)+2^d)$	$2d$	AHE, OT
Tueno[17](OT)	d	$O((n+m)l)$	$4d$	SS, OT
Tueno[17](GC)	d	$O((n+m)l)$	$4d$	SS, GC, OT
Tueno[17](ORAM)	d	$O(d^4 l)$	d^2+3d	SS, ORAM, GC
Ma[18]	d	$O(dnl)$	$2d-1$	SS, GC, OT

TABLE II
NOTATION DESCRIPTIONS

Notations	Definitions
λ	Statistical analysis of security parameters
m	Number of decision nodes in a decision tree
d	Depth of decision tree
d'	Predefined depth of decision tree
T	Decision tree
$\boldsymbol{X}=(x_1,...,x_n)$	Feature vector of length n
n	Number of feature values in the user's vector
A_T	Encoding array of decision tree
l/r	Index of left child/right child
t	Threshold value of node
v	Index of user feature
c	Predicted value of leaf node
ℓ	Bit length of default Boolean sharing
ℓ_v	Length of decision tree array element
ℓ_b	Bit length of pseudorandom function output

node contains an attribute value and a branch, while each leaf node represents a category or a numerical value.

The construction of a decision tree typically involves selecting the optimal attributes to split the dataset, maximizing the separability of each resulting subset. To achieve this, decision trees employ various metrics to evaluate attribute importance, including information gain, Gini impurity, and gain ratio.

In the process of constructing decision trees, several algorithms can be utilized to determine the sequence of decision nodes and splitting points, including ID3, C4.5, CART, and others. The primary distinctions among these types of decision trees lie in their sensitivity to attribute values, which refers to the degree of variability or uniqueness of a given attribute across different samples in the dataset.

B. Cryptographic Primitives

We provide a brief overview of the cryptographic primitives used in SS-PDI and summarize the notations in Table II.

1)arithmetic Secret Sharing (ASS)

Specifically, for data $x \in \mathbb{Z}_{2^\ell}$, where $\mathbb{Z}_{2^\ell}$ denotes a ring with a modulus of 2^ℓ agreed upon by the parties, an arithmetic secret can be decomposed into shares. This decomposition is represented as $\langle x \rangle = \langle x \rangle_0 + \langle x \rangle_1$, where $\langle \cdot \rangle_i$ signifies the share held by party P_i (i $\in \{0, 1\}$).

Some basic computation protocols are described as follows:

- Secret Sharing(Shr): The secret $x \in \mathbb{Z}_{2^\ell}$ is owned by party P_0, who generates a random number $r \in \mathbb{Z}_{2^\ell}$. r is then sent to the party P_1, x is split into two parts: $\langle x \rangle_0 = x - r$ and $\langle x \rangle_1 = r$, denote $\langle x \rangle = (\langle x \rangle_0, \langle x \rangle_1)$.
- Secret Restore(Rec): The secret shares $\langle x \rangle_i$ are merged to reveal the secret x. P_1 sends the $\langle x \rangle_1$ to x owner P_0, who restores the x by computing $\langle x \rangle_0 + \langle x \rangle_1$, denote Restore($\langle x \rangle$).
- Secure Addition: Party $P_i, i \in \{0,1\}$ holds $\langle x \rangle_i$ and $\langle y \rangle_i$ and then P_i computes $\langle x \rangle_i + \langle y \rangle_i$ to obtain shares of $\langle x+y \rangle_i$, denote SecAdd($\langle x \rangle_i, \langle y \rangle_i$).
- Secure Multiplication: Party $P_i, i \in \{0,1\}$ holds two multiplicands $\langle x \rangle_i$ and $\langle y \rangle_i$. With the help of Beaver's Triple to achieve secure Multiplication, where the triple $\{u, v, k\}$ satisfies $k = uv$, and the triple is shared between P_i. P_i computes $\langle d \rangle_i = \langle x \rangle_i - \langle u \rangle_i$ and $\langle e \rangle_i = \langle y \rangle_i - \langle v \rangle_i$. Then P_i Restore d and e. P_0 compute $\langle z \rangle_0 = \langle k \rangle_0 + e\langle u \rangle_0 + d\langle v \rangle_0 + de$ and P_1 compute $\langle z \rangle_1 = \langle k \rangle_1 + e\langle u \rangle_1 + d\langle v \rangle_1$ to obtain the shares of $\langle z \rangle_i$, where z satisfy $z = xy$, denote SecMul($\langle x \rangle_i, \langle y \rangle_i$).

In order for cryptographic primitives to be compatible with the secure neural network inference, they have to support arithmetic operations on shared decimal numbers.References to work in [27], We transform the fixed-point decimal numbers x and y (with at most ℓ_F bits in the fractional part) to integers by letting $x' = 2^{\ell_F} x$ and $y' = 2^{\ell_F} y$. To perform multiplication of x and y, we multiply x' and y' to get the product $z = x'y'$ and truncate the last ℓ_F bits of z such that it still has ℓ_F bits representing the fractional part.

2)Boolean Secret Sharing(BSS)

Boolean secret sharing (BSS) can be regarded as arithmetic sharing in $\mathbb{Z}_2$ in which the addition operation is replaced by the XOR operation ($\oplus$) and multiplication is replaced by the AND operation. So their corresponding Shr and Rec algorithms can be defined in a similar manner as the arithmetic sharing.

The functionality of B2A protocol (for boolean sharing to arithmetic sharing conversion) takes boolean shares as input and gives out arithmetic shares of the same value as output. We refer the readers to [28] for details of Boolean sharing and B2A protocol.

IV. SYSTEM OVERVIEW

In this section, we describe the overall framework of the system, the types of participants and the roles each plays.

The entire system contains the following four entities:

- **Client (C)**: Client C shares its feature vector information through arithmetic secret sharing to generate a pair of messages $\langle \boldsymbol{X} \rangle_0$ and $\langle \boldsymbol{X} \rangle_1$, which are respectively sent to cloud servers S_0 and S_1 without disclosing any messages to other entities.
- **Model Provider (P)**: P represents its trained decision tree model T as an array A_T and also performs additive secret sharing to generate a pair of messages $[A_T]_0$ and$[A_T]_1$, which are respectively sent to S_0 and S_1 before entering the offline state.
- **Cloud Server (S_0)**: S_0 is an honest-but-curious participant with powerful storage and computing capabilities. Which means that while S_0 strictly follows the protocol, it may attempt to infer private information of other parties during the execution process. S_0 receives the decision tree model from the model provider and the feature vector from the client. It interacts with another cloud server to perform secure outsourcing of task computation.
- **Cloud Server (S_1)**: S_1 is an honest-but-curious participant with powerful storage and computing capabilities, and has the same role as S_0.

The entire system consists of the model provider, the client (C), and two cloud servers(S_0, S_1). The model provider (P) deploys its model onto the two cloud servers using secret sharing, ensuring that the model remains confidential. Similarly, the client also secret shares owned data and sends it to the two cloud servers. These two cloud servers form a dual-cloud outsourcing system, where they interact with each other to process the data uploaded by the client. Finally, each cloud server returns the processed results, which are in the form of secret shares, to the client. This architecture ensures that both the model and the client's data remain secure throughout the entire process, with no party having full access to the sensitive information.

V. Methods

In this section, we will introduce the system runtime process is divided into four phases, and each phase of both severs' execution of the action and the results of the phase run, respectively.

A. Prepare Phase

The classification model represented by the decision tree, denoted as T, is structured as a tree comprising a root node, internal nodes, and leaf nodes. Each internal node contains a threshold value t, indices for the left and right child nodes, ℓ and r, respectively, and a corresponding feature vector index v. The structure of the root node and internal nodes is identical. In contrast, the leaf node stores the prediction result c of the decision tree. The nodes are sequentially numbered starting from the root node, and are labeled from left to right, layer by layer, with v_0 representing the root node. The decision tree T is represented as an array $A_T = [A_T[0], \ldots, A_T[m-1]]$, where m denotes the total number of nodes. The nodes are indexed as T_v, where $v \in \{0, 1, \ldots, m-1\}$. The user provides a feature vector $X = (x_0, \ldots, x_{n-1})$, where n is the number of feature values in the user's vector, and the elements are denoted as X_v, where $v \in \{0, 1, \ldots, n-1\}$.

The improved structure of the decision tree, as compared to the traditional one, is illustrated in Fig. 1. In this enhanced structure, the left and right child node indices of the leaf nodes both point to themselves. This modification keeps the number of comparison operations required constant, hiding the access patterns. Specifically, the number of comparison operations for all prediction outcomes is solely dependent on the depth of the decision tree. As a result, attackers are unable to infer the structure of the decision tree based on the number of comparison operations. The decision tree is encoded in an array format, with nodes numbered in depth-first traversal order, as shown in Fig. 2. The tree is composed of five data items: 1) node threshold t; 2) left child index ℓ; 3) right child index r; 4) feature vector ID v; and 5) predicted label c. In the figure, the symbol ▲ represents any assignable value, while ★ denotes a random value selected from the range $[0, 1, \ldots, n-1]$, where n is the number of feature values in the user's feature vector.

B. Feature Selection Phase

This section involves two rounds of feature selection: one for selecting the client feature vector array and another for selecting the decision tree array. First, we discuss the selection of the feature vector array. Upon receiving the shared client feature vectors, the cloud server performs secure feature selection by securing the feature vector array and generating secret shares for the corresponding feature values based on the feature ID in the root node of the decision tree. To facilitate the selection of the feature vector array, a secure feature selection function is designed, which takes as input the shared secret shares of the client feature vector $\boldsymbol{X}$ and the feature ID $\langle \mathcal{I} \rangle$, and outputs the selected shared shares $\langle \boldsymbol{X}[\mathcal{I}] \rangle$. This process is denoted as $F^X_{SFS}(\mathcal{I}_0, \mathcal{I}_1) \rightarrow (\langle \boldsymbol{X}[\mathcal{I}] \rangle_0, \langle \boldsymbol{X}[\mathcal{I}] \rangle_1)$.

To achieve the selection of the client feature vector array, our scheme employs the Secure Feature Selection Protocol SFS, utilizing 1-out-of-n OT [17] during the feature selection phase. We also designs a secure feature selection scheme under a dual-cloud architecture, as described in Algorithm 1. Each cloud server S_b ($b \in \{0, 1\}$) first generates a random value r_b, then shifts the index of the vecotr $\langle \boldsymbol{X} \rangle_b$ by sharing the corresponding share $[\mathcal{I}]_b$, and exports a new vector $\{\boldsymbol{E}_i\}_b = r_b \oplus \boldsymbol{X}_b[i + [\mathcal{I}]_b \pmod m]$, after performing an XOR operation with the random value r_i. This new array is then sent to the other cloud server for XOR calculation, resulting in the target shared value. For instance, taking receiver S_1 as an example, the new array $\{\boldsymbol{E}_i\}_0$ is received. Since only S_0 knows the index value for the array shift and the new array is masked by the random value r_0, receiver S_1 cannot retrieve the value of the other share of the array, ensuring the security of the process.

C. Security Comparison Phase

In the dual-cloud model, feature selection algorithms are independently applied to the user feature vector $\boldsymbol{X}$ and the decision tree array A_T. When the feature selection target

Algorithm 1 Selection of feature vector

Require: S_b input $\langle \mathcal{I} \rangle_b$, $b \in \{0,1\}$
Output: S_b output $\langle \boldsymbol{X}[\mathcal{I}] \rangle_b$, $b \in \{0,1\}$
1: Initialize array length n; index bit length l; and array item bit length l_v.
2: S_0 and S_1 run B2A to convert $\langle \mathcal{I} \rangle$ to $[\mathcal{I}] \in Z_n$;
3: **for** each S_b execute **do**
4: $r_b \leftarrow Z_n^{l_v}$;
5: $\{E_i\}_b = r_b \oplus \boldsymbol{X}_b[i + [\mathcal{I}]_b (mod\ n)] \quad i \in [0, n)$;
6: S_b and $S_{(b+1)\ mod\ 2}$ call 1-out-of-n OT function F_{ot}:
7: S_b input $\{E_i\}_b$ and $S_{(b+1)\ mod\ 2}$ obtaining $r_b \oplus \boldsymbol{X}_{(b+1)\ mod\ 2}[\mathcal{I}]$
8: $\langle \boldsymbol{X}[\mathcal{I}] \rangle_b \leftarrow r_b \oplus r_{(b+1)\ mod\ 2} \oplus X_{(b+1)\ mod\ 2}[\mathcal{I}]$;
9: S_b output $\langle \boldsymbol{X}[\mathcal{I}] \rangle_b$

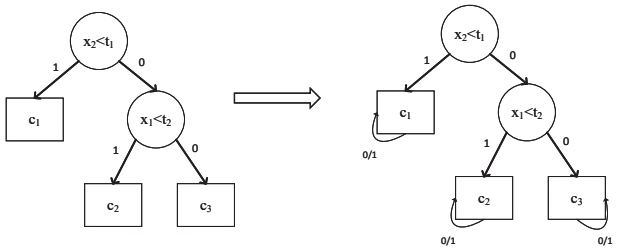

Fig. 1. Improvement of Decision Tree Data Structure.

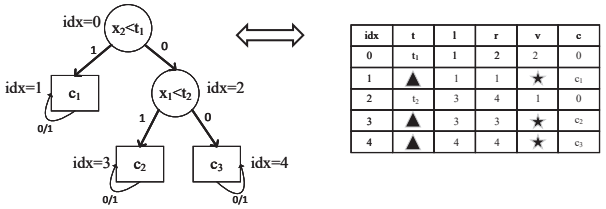

Fig. 2. Decision Tree Array Encoding.

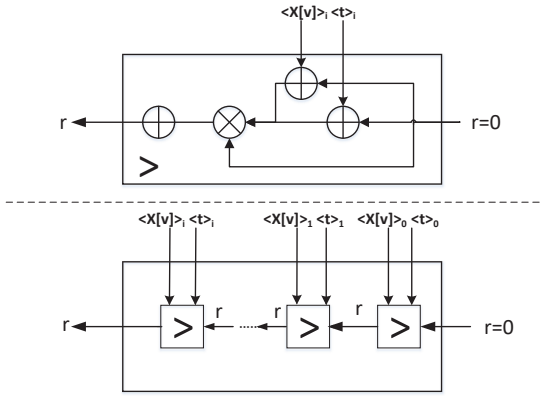

Fig. 3. Comparison Boolean Circuit.

M is the user feature vector $\langle v \rangle$, the dual-cloud executes F_{SFS}^{M}, yielding Boolean shares $\langle \boldsymbol{X}[v] \rangle_0$ and $\langle \boldsymbol{X}[v] \rangle_1$. When M corresponds to the decision tree, the dual-cloud executes $F_{SFS}^{A_T}$ based on the index $\langle v \rangle$, resulting in Boolean shares $\langle A_T[\mathcal{I}] \rangle_0$ and $\langle A_T[\mathcal{I}] \rangle_1$. Since $A_T[\mathcal{I}] \rightarrow t||l||r||v||c$, the dual-cloud only needs to compare the node thresholds during the comparison phase. Specifically, the node thresholds $\langle [t] \rangle_0$ and $\langle [t] \rangle_1$ are derived from parsing the Boolean shares.

After this process, cloud servers S_0 and S_1 each hold secret shares of the feature vector $\langle \boldsymbol{X}[v] \rangle$ and the decision node $\langle A_T[\mathcal{I}] \rangle$, respectively. Subsequently, the two clouds need to securely compare $\langle A_T[\mathcal{I}] \rangle$ with the corresponding $\langle \boldsymbol{X}[v] \rangle$. If $\langle A_T[\mathcal{I}] \rangle < \langle \boldsymbol{X}[v] \rangle$, the comparison result b is set to 1; otherwise, b is set to 0. Existing schemes [8], [9], [29] typically rely on expensive obfuscation circuits or homomorphic encryption to perform secure size comparisons. To reduce the computational overhead of these comparisons, this chapter's solution employs a binary representation of values, converting the size comparison operation into a Most Significant Bit (MSB) operation.

Specifically, this work adopts a privacy-preserving comparison method based on Boolean sharing [30]. The basic principle involves utilizing the classic Goldreich-Micali-Wigderson (GMW) protocol to determine the Boolean circuit that facilitates the comparison operation. This circuit is constructed using the fixed-length (length ℓ_b) Boolean shared attribute values $\langle \boldsymbol{X}[v] \rangle$ and node thresholds $\langle [t] \rangle$. The protocol specifies the operations that each server (S_0 or S_1) must perform. Throughout the execution of the circuit, no complex ciphertext operations are involved, which significantly enhances the efficiency of the secure comparison. As shown in Fig. 3, when implementing comparison operation with boolean circuit, two operations are involved: "XOR" operation $\oplus$ and "AND" operation $\otimes$. Based on the above comparison circuit and the dissimilarity and sum operations in the context of Boolean sharing, Algorithm 2 shows the process of implementing a secure comparison between cloud servers S_0 and S_1 based on their own Boolean sharing shares.

Algorithm 2 Privacy-preserving boolean comparison

Require: Cloud server S_0 inputs $\langle \boldsymbol{X}[v] \rangle_0$,$\langle t \rangle_0$,auxiliary bit r_0; Cloud serverS_1 inputs $\langle \boldsymbol{X}[v] \rangle_1$, $\langle t \rangle_1$ and r_1, where $r_0 \oplus r_1 = 0$.
Output: S_0 obtains the comparison result $\langle b \rangle_0$, S_1 obtains $\langle b \rangle_1$
1: **for** each $i \in [1, \ell_b]$ **do**
2: S_0:$r_0 = (r_0 \oplus \langle \boldsymbol{X}[v] \rangle_{0,i}) \otimes (r_0 \oplus \langle t \rangle_{0,i}) \oplus r_0$;
3: S_1:$r_1 = (r_1 \oplus \langle \boldsymbol{X}[v] \rangle_{0,i}) \otimes (r_1 \oplus \langle t \rangle_{1,i}) \oplus r_1$;
4: S_0 obtains$\langle b \rangle_0 \leftarrow r_0$;
5: S_1 obtains$\langle b \rangle_1 \leftarrow r_1$;

D. Evaluation Phase

After the secure comparison calculation, the dual-cloud servers obtain the shared comparison result $\langle b \rangle_i$. This comparison result determines whether the decision node selects the left or right child node as the next decision node. Specifically, the next index is computed using the Equation 1.

$$\langle \mathcal{I} \rangle \leftarrow (\langle l \rangle + \langle b \rangle) \cdot (\langle l \rangle + \langle r \rangle) \tag{1}$$

where if $b = 1$, the next index $\langle \mathcal{I} \rangle$ is set to $\langle l \rangle$; if $b = 0$, the index is set to $\langle r \rangle$.

The entire prediction result generation process follows Algorithm 3. The loop is iterated d times, and during each iteration, the servers S_0 and S_1 call the function $F_{SFS}^{(\boldsymbol{X})}$ to obtain the secret shares of $\boldsymbol{X}[v]$, then perform the secure comparison process to compute the comparison result b. The comparison result $\langle b \rangle$ is then used by S_0 and S_1 to determine the next decision tree index $\langle \mathcal{I} \rangle$. Following this, S_0 and S_1 call the function $F_{SFS}^{(A_T)}$ based on the new index $\langle \mathcal{I} \rangle$ to obtain $\langle A_T[\mathcal{I}] \rangle$, and finally parse $\langle c \rangle$ as the prediction result $\langle r \rangle$.

Due to the design of the self-loop in the leaf nodes of the binary tree, when the loop reaches a leaf node, it executes a self-loop that points to itself. Thus, $\langle r \rangle$ becomes the final target result. Moreover, the self-loop design hides the number of comparison operations, ensuring that all prediction results are only revealed after d iterations. The client C obtains the secret shares $\langle r \rangle_0$ and $\langle r \rangle_1$ from S_0 and S_1, and computes the final prediction result.

Algorithm 3 Prediction Result Generation

Require: $A_T[0]$
Output: S_b obtains comparison result $\langle r \rangle_b$

S_0 and S_1 share $A_T[0]$ and jointly parse $\langle A_T[0] \rangle$ into $\langle t \rangle || \langle l \rangle || \langle r \rangle || \langle v \rangle || \langle c \rangle$
for each $b \in \{0, 1\}$ **do**
    S_b sends $(Eval, \langle v \rangle_b)$ to $F_{SFS}^{(\boldsymbol{X})}$, S_b share $\langle \boldsymbol{X}[v] \rangle_b$;
    S_b perform secure comparison $\langle b \rangle_b \leftarrow \langle \boldsymbol{X}[v] \rangle_b > \langle t \rangle_b$;
    S_b calculate the index of the next decision node $\langle \mathcal{I} \rangle_b \leftarrow (\langle l \rangle_b + \langle b \rangle_b) \cdot (\langle l \rangle_b + \langle r \rangle_b)$;
    S_b sends $(Eval, \langle \mathcal{I} \rangle_b)$ to $F_{SFS}^{(A_T)}$, S_b jointly share $\langle A_T[\mathcal{I}] \rangle_b$;
    Parse $\langle t \rangle || \langle l \rangle || \langle r \rangle || \langle v \rangle || \langle c \rangle \leftarrow A_T[\mathcal{I}]_b$;
    $\langle r \rangle_b \leftarrow \langle c \rangle_b$;
S_b obtain the shared result $\langle r \rangle_b$.

VI. THEORETICAL ANALYSIS

In this section, we first analyze the security of our proposed protocols, and then analyze their computation cost and communication overhead.

A. Security Analysis

In this chapter, the participants are all assumed to be semi-honest. If the scheme is secure under the semi-honest model, the participants are only able to access the data and public parameters specified by the protocol during the secure outsourcing of the decision tree, and they cannot infer any arbitrary messages related to the original inputs. The security of the system is analyzed in three distinct scenarios, each corresponding to a semi-honest attacker capturing one of the parties: P, C, or S.

When the model provider P is the attacker,since P only provides the secret sharing of the decision tree to the cloud server and does not receive any messages during the protocol execution, the attacker P does not gain access to any additional information. This confirms the security of the scheme in this scenario.

When the client C is the attacker, the client C only provides the secret-shared values of the feature vector $\boldsymbol{X}$ and receives the two shares of the prediction results, $\langle r \rangle_0$ and $\langle r \rangle_1$. The client does not participate in the decision-making process of the prediction, and as a result, does not obtain any additional information beyond what is explicitly shared. Thus, the security of the scheme is maintained even in the presence of a client adversary.

When the cloud service provider S is the attacker, the roles of the two cloud servers, S_0 and S_1, are symmetric. It is assumed that the two cloud servers are not colluding, and therefore, this section analyzes the case where only a single cloud server, S_0, is the attacker. The inputs and outputs of the cloud servers are related to the secret sharing of data, and the security of additive secret sharing ensures that the shared data remains random and protected. The interaction between the cloud servers at different stages of the protocol is analyzed as follows:

1) Preparation Phase: The feature vector and decision tree array are abstracted into a message M. Cloud server S_0 holds the secret-shared value M_0 and applies a random function $F(sk_0, \mathcal{I})$ to obtain $M_0[\mathcal{I}] \oplus F(sk_0, \mathcal{I})$. Cloud server S_1 receives this perturbed value and cannot decrypt it to obtain the original plaintext. Cloud server S_1 then adds $M_1[\mathcal{I}] \oplus F(sk_1, \mathcal{I})$ to the received perturbed value and sends the result back to S_0. Since the final plaintext is obscured by the random values, cloud server S_0 is also unable to retrieve the original data.
2) Feature Selection Phase: By the relevant results in [31]. it is known that if the function F is a secure pseudo-random function, then the oblivious selection protocol in Algorithm 1 can securely compute the function $F_{SOS}^{(M)}$ in the hybrid model (F_{sprf}, F_{pre}) under the 'honestly curious' adversary model.
3) Secure Comparison Phase: The secure comparison phase utilizes Boolean sharing-based privacy comparison. The 'iso-operation' of the secure comparison circuit is executed locally on the cloud server without any interaction. The other operations, which involve interaction between the two cloud servers, are securely executed via the BMT triplet protocol [32], ensuring the integrity and confidentiality of the comparison process.
4) Evaluation Phase In the evaluation phase, the cloud server only receives the secret-shared values of the predicted results. A single shared value is effectively a random value for the cloud server, and thus, it cannot infer the prediction. Two shared values must be combined to recover the final result, ensuring the security of this phase.

B. Performance Analysis

Let n be the number of attributes of the feature vector, l be the number of binary bits per feature value, d be the depth of the decision tree, m be the number of nodes in the decision tree, OT represent the inadvertent transmission protocol, MT represent the number of Beaver's multiplicative triples, SS

denote secret sharing, and r_F denote the number of rounds to execute the pseudo-random function (PRF). This section analyzes the scheme proposed in this section in four phases:

- Preparation Phase: In the preparation phase, m decision nodes are secret-shared, with 5 mt plaintexts containing their respective subscript random values. These values must be dissociated to obtain the secret text C_T (the 5 items include the node threshold, left child index, right child index, etc.). Each of these items consists of binary bits. The overhead in this phase can be handled offline by the user, so we focus only on the online overhead required when the user initiates a query.
- Selection Phase: In the selection phase, a decision node and the corresponding user feature vector need to be selected for comparison. For feature vector selection, the 1-out-of-n OT function is used. This requires the sender to transmit the entire user feature array of size nl to the receiver. Additionally, the selection of the decision tree node involves generating 5ℓ random values for interaction. Therefore, the communication complexity of the selection phase is $O((n + 5m)\ell)$.
- Comparison Phase: In the comparison phase, a comparison operation requires l multiplications. To achieve this, l Beaver's multiplicative triples are needed for secure multiplication.
- Evaluation Phase: In the evaluation phase, the prediction results are represented in the form of secret sharing. Only a single secret recovery operation is required at the end of the phase.

Table III compares the overhead of each phase in the proposed scheme with the schemes from the literature [12] and [31]. Since the above analysis pertains to a single process, the final complexity must be multiplied by the number of comparisons d.

Compared to the scheme in the literature [12], our approach requires fewer comparisons. In comparison with the scheme in [31], our scheme utilizes the GMW protocol [33] to complete the comparison circuit, while the GC protocol is used in [31]. However, the communication overhead in [31] is $O(dnl)$, which is smaller than that in our scheme. This is because the scheme in [31] is based on a Client-Server (CS) architecture, where the decision tree is held by a single participant, and the communication overhead is only dependent on the 1-out-of-n OT function. Nonetheless, due to the differences in architecture, the complexity of the selection phase in our scheme is still acceptable.

VII. Experiment

A. Implementation and Experimental Setup

This paper evaluates the efficiency of the scheme presented in this section using C++. The implementation of the secret sharing technique is based on the ABY framework library [34], while the SFS protocol is implemented using Floram [35]. The LowMC technique is utilized to instantiate the PRF function and the ASS and SMPC (Secure Multi-Party Computing, a cryptographic protocol that enables multiple parties to collaboratively compute a function over their private inputs with no party gains any information about the others' inputs beyond what is revealed by the output) related code implementation is referenced from NssMPClib [36]. In the experiment, the computational security parameter is set to $k = 128$, and the statistical security parameter is set to $\lambda = 40$. The algorithm used in constructing decision trees is CART.

The experiment is conducted within a Docker container on an Ubuntu 18.04 system. Two containers, simulating two cloud servers, are configured with AMD EPYC Milan processors and run on a Tencent Cloud virtual machine with 4GB of RAM. Two network environments are simulated: a local area network (LAN) and a wide area network (WAN).

In the LAN environment, the S_0 and S_1 server instances are deployed within the same server room. In the WAN environment, the S_0 and S_1 server instances are located in Beijing Area 2 and Guangzhou Area 6, respectively. To measure the network performance, we use the 'ping' command to evaluate network latency and the 'iperf' command to measure the network bandwidth under both network conditions. The network latency and bandwidth values are summarized in the TABLE IV.

The experiments were conducted using five publicly available datasets from UCI (University of California Irvine) [37]: Heart-disease, Credit-screening, Breast-cancer, Housing, and Spambase. These datasets are used to evaluate the performance of the proposed scheme. The focus of the experiments is on three key characteristics of the datasets: feature vector dimension n, decision tree depth d, and number of decision nodes m, as shown in TABLE V.

B. Performance Evaluation

In this section, we experiment with the five datasets mentioned above and compare the results against similar schemes from the literature, namely [12] and [11], which focus on decision trees based on dual-cloud architectures. Specifically, [12] targets optimizing decision trees over wide-area networks (WAN), while [11] evaluates secure decision trees in a local-area network (LAN) environment. Since [12] was tested solely in a WAN setup, and [11] was tested in a standalone environment, the results from both schemes are shown in the table for comparison.

All three schemes—SS-PDI, [12], and [11]—are based on a dual-cloud architecture, where both the client and the cloud server go offline after the initial data transmission to the cloud server. As a result, the computational overhead during the offline phase is similar across all schemes. Therefore, only the running time of the inference phase, executed on the cloud server, is considered. TABLE VI presents a comparison of the computational overhead of the SS-PDI scheme in both LAN and WAN network environments, relative to the schemes in [11] and [12].

The comparison results reveal that in the LAN network environment, the computational overhead of our scheme is higher than that of the scheme from [11] for the first four datasets. However, for the 'deep and sparse' Spambase dataset, the proposed scheme outperforms the [11] protocol, with the latter being 3.98 times faster in terms of computational

TABLE III
SS-PDI PERFORMANCE COMPARISON

Scheme	Number of comparisons	Selection phase	Comparison phase	Evaluation phase	Communication rounds
Scheme in [12]	2^d	$m\binom{1}{n}OT$	$O(m2^l \cdot MT)$	m·SS	$(\log_2 l + 5)d$
Scheme in [31]	d	$O(dnl)$	dGC	SS	$(3r_F + 5)d$
Our scheme	d	$O(d(n+5m)l)$	$O(dl \cdot MT)$	SS	$(3r_F + 3 + l)d$

TABLE IV
TEST NETWORK ENVIRONMENT

Network Evironment	Delay	Bandwidth
LAN	1-3ms	1Gbps
WAN	60-80ms	4Mbps

TABLE V
RELEVANT FEATURES OF THE DECISION TREE DATASET

Dataset	d	m	n
Heart-disease	3	5	13
Credit-screening	4	5	15
Breast-cancer	8	12	9
Housing	13	92	13
Spambase	17	58	17

overhead. However, in the WAN network environment, our scheme exhibits an overhead that is approximately the same as in the LAN, significantly lower than that in [12]. This demonstrates the strong stability of our scheme, as it maintains consistent performance across different network environments, unlike previous works where the overhead may increase substantially in a wide area network. This stability highlights the robustness of our privacy-preserving approach, making it well-suited for real-world deployment in diverse network conditions. Additionally, our scheme exhibits a much smoother runtime variation across different datasets, which is attributable to our strict adherence to privacy-preserving rule design protocols that effectively obscure data access patterns. This ensures that the computational overhead remains more consistent, regardless of dataset characteristics.Since both [11] and [12] were proposed by Zheng et al., these two protocols will be collectively referred to as the 'Zheng protocol' in the subsequent analysis.

In our opinion, There are two main shortcomings in the Zheng's protocol:

1) The protocol requires execution under the assumption of a complete binary tree. If this assumption is violated, there is a security risk where the tree structure could potentially be inferred based on the number of comparisons.
2) The protocol involves comparing all decision nodes, with the comparison process requiring the execution of m instances of $F_{1-out-of-n}$ OT operations.

In the case of the Spambase dataset, with a feature vector size $n = 57$ and decision tree depth $d = 17$, the decision tree is extended to a complete binary tree, resulting in an exponential increase in query time. In contrast, the scheme proposed in this section only requires d comparisons, making it more suitable for sparse binary trees and resulting in a lower computational overhead than the Zheng protocol for the "deep and sparse" Spambase dataset.

We also compared it to other programs, as shown in TABLE VII. The evaluation is conducted against the existing solution proposed by Doerner et al. [38]. The results, as shown in Table VII, encompass three distinct phases: Security Feature Selection, Security Node Comparison, and Secure Inference Generation. For each phase, we compare the runtimes of both the SS-PDI solution and the approach presented by Doerner et al. across several widely used datasets.

Similar to the previous experimental results, the improvement in execution efficiency of SS-PDI on general datasets is not significant However, on deep and sparse datasets, SS-PDI demonstrates a significant advantage over traditional schemes. This advantage arises from the scheme's ability to handle sparse decision trees more efficiently, leading to reduced computational overhead. Furthermore, our privacy-preserving scheme is simpler compared to previous work, offering an elegant solution that minimizes the complexity of operations while maintaining strong security guarantees.

Overall, the comparison highlights the trade-off between security and performance. While the SS-PDI solution incurs higher runtime costs in certain phases, its enhanced security features could make it a more suitable choice for sensitive applications, where secure computation is paramount. The results also suggest areas for future optimization, particularly in reducing the runtime in the Security Feature Selection and Security Node Comparison phases without compromising the security guarantees.

VIII. CONCLUSION

In this paper, we address the problem that existing security outsourcing decision tree inference services cannot balance security and computational efficiency in large decision trees, a privacy decision tree inference scheme based on secret sharing in a dual cloud architecture is proposed. Using oblivious transfer and secret sharing techniques, the feature selection module is improved to reduce the number of feature selections, and the computational overhead and communication overhead of feature selection sublinearity are achieved. Based on the binary tree characteristics, the decision tree parent and child nodes are packed and stored in a cluster of two-dimensional arrays to reduce the number of times to perform decision tree SOS from d times to d/q times. A secure comparison circuit is designed using Boolean sharing to reduce the overhead of secure comparison. In the semi-honest threat environment, the security of the system when the model provider, client, and cloud server are the attackers is analyzed, and the experimental analysis proves that the scheme in this section has smaller

TABLE VI
COMPUTATIONAL OVERHEAD OF SS-PDI SCHEME COMPARED WITH RELATED SCHEMES

Net type	Scheme	Heart-disease	Credit-screening	Breast-cancer	Housing	Spambase
LAN	Scheme in [11]	0.0013	0.0029	0.0034	0.132	6.7
	SS-PDI	0.724	0.781	0.932	1.425	1.681
WAN	Scheme in [12]	1.21	1.37	1.45	3.33	27.07
	SS-PDI	0.862	0.926	1.238	1.410	16.02

TABLE VII
COMPARISON OF RUNTIME OF SS-PDI SOLUTION FOR DIFFERENT PHASES OF CLOUD SERVER EXECUTION IN WAN NETWORK ENVIRONMENT

Dataset	Security Feature Selection		Security Node Comparison		Secure Inference Generation
	Scheme in [38]	SS-PDI	Scheme in [38]	SS-PDI	Scheme in [38]
Heart-disease	0.527	0.413	0.529	0.214	0.154
Credit-screening	0.529	0.435	0.53	0.257	0.306
Breast-cancer	0.533	0.517	0.532	0.305	0.383
Housing	0.91	0.662	1.735	0.352	0.69
Spambase	21.006	11.994	4.281	5.540	1.785

computational overhead and communication overhead when executing decision tree inference for large decision trees, which satisfies the basic requirements of secure decision tree inference.

REFERENCES

[1] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami, and R. R. Zebari, "Iot and cloud computing issues, challenges and opportunities: A review," *Qubahan Academic Journal*, vol. 1, no. 2, pp. 1–7, 2021.

[2] J. Liang, Z. Qin, S. Xiao, L. Ou, and X. Lin, "Efficient and secure decision tree classification for cloud-assisted online diagnosis services," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1632–1644, 2019.

[3] B. W. Yap, S. H. Ong, and N. H. M. Husain, "Using data mining to improve assessment of credit worthiness via credit scoring models," *Expert Systems with Applications*, vol. 38, no. 10, pp. 13 274–13 283, 2011.

[4] Z. Xu, L. Zhao, W. Liang, O. F. Rana, P. Zhou, Q. Xia, W. Xu, and G. Wu, "Energy-aware inference offloading for dnn-driven applications in mobile edge clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 799–814, 2020.

[5] H. Ye, X. Zhang, Z. Huang, G. Chen, and D. Chen, "Hybriddnn: A framework for high-performance hybrid dnn accelerator design and implementation," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.

[6] H. Tabrizchi and M. Kuchaki Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *The journal of supercomputing*, vol. 76, no. 12, pp. 9493–9532, 2020.

[7] A. Masood, D. S. Lakew, and S. Cho, "Security and privacy challenges in connected vehicular cloud computing," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2725–2764, 2020.

[8] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *Cryptology ePrint Archive*, 2014.

[9] D. J. Wu, T. Feng, M. Naehrig, and K. Lauter, "Privately evaluating decision trees and random forests," *Cryptology ePrint Archive*, 2015.

[10] R. K. Tai, J. P. Ma, Y. Zhao, and S. S. Chow, "Privacy-preserving decision trees evaluation via linear functions," in *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22*, 2017, pp. 494–512.

[11] Y. Zheng, H. Duan, C. Wang, R. Wang, and S. Nepal, "Securely and efficiently outsourcing decision tree inference," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1841–1855, 2020.

[12] Y. Zheng, C. Wang, R. Wang, H. Duan, and S. Nepal, "Optimizing secure decision tree inference outsourcing," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[13] C. Guo, K. Cheng, J. Fu, R. Fan, Z. Chang, Z. Zhang, and A. Song, "Gfs-cnn: A gpu-friendly secure computation platform for convolutional neural networks," *Journal of Networking and Network Applications*, vol. 3, no. 2, pp. 66–72, 2023.

[14] A. Song, J. Fu, X. Mu, X. Zhu, and K. Cheng, "L-secnet: Towards secure and lightweight deep neural network inference," *Journal of Networking and Network Applications*, vol. 3, no. 4, pp. 171–181, 2024.

[15] M. Joye and F. Salehi, "Private yet efficient decision tree evaluation," in *Data and Applications Security and Privacy XXXII: 32nd Annual IFIP WG 11.3 Conference, DBSec 2018, Bergamo, Italy, July 16–18, 2018, Proceedings 32*, 2018, pp. 243–259.

[16] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *Information Security and Privacy: 12th Australasian Conference*, 2007, pp. 416–430.

[17] A. Tueno, F. Kerschbaum, and S. Katzenbeisser, "Private evaluation of decision trees using sublinear cost," *Proceedings on Privacy Enhancing Technologies*, vol. 1, 2019.

[18] J. P. Ma, R. K. Tai, Y. Zhao, and S. S. Chow, "Let's stride blindfolded in a forest: Sublinear multi-client decision trees evaluation." in *NDSS*, 2021.

[19] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," *Cryptology ePrint Archive*, 2014.

[20] Y. Ishai and A. Paskin, "Evaluating branching programs on encrypted data," in *Theory of Cryptography Conference*. Springer, 2007, pp. 575–594.

[21] L. Liu, J. Su, R. Chen, J. Chen, G. Sun, and J. Li, "Secure and fast decision tree evaluation on outsourced cloud data," in *Machine Learning for Cyber Security: Second International Conference*, 2019, pp. 361–377.

[22] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. Nascimento, W.-S. Poon, and S. Truex, "Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 2, pp. 217–230, 2017.

[23] G. Kiss, M. Naderpour, J. Liu, N. Asokan, and T. Schneider, "Sok: Modular and efficient private decision tree evaluation," *Proceedings on Privacy Enhancing Technologies*, vol. 2, 2019.

[24] Y. Zheng, H. Duan, and C. Wang, "Towards secure and efficient outsourcing of machine learning classification," in *Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part I 24*. Springer, 2019, pp. 22–40.

[25] K. Ji, B. Zhang, T. Lu, L. Li, and K. Ren, "Uc secure private branching program and decision tree evaluation," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 2836–2848, 2022.

[26] J. Fu, K. Cheng, Y. Xia, A. Song, Q. Li, and Y. Shen, "Private decision tree evaluation with malicious security via function secret sharing," in *European Symposium on Research in Computer Security*. Springer, 2024, pp. 310–330.

[27] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE symposium on security and privacy (SP)*, 2017, pp. 19–38.
[28] D. Demmler, T. Schneider, and M. Zohner, "Aby-a framework for efficient mixed-protocol secure two-party computation." in *NDSS*, 2015.
[29] S. S. Sathya, P. Vepakomma, R. Raskar, R. Ramachandra, and S. Bhattacharya, "A review of homomorphic encryption libraries for secure computation," *arXiv preprint arXiv:1812.02428*, 2018.
[30] L. Ma, J. Peng, Q. Pei, and H. Zhu, "Efficient decision tree privacy classification service protocol," *Journal On Communications*, vol. 42, no. 8, pp. 80–89, 2021.
[31] J. Bai, X. Song, S. Cui, E.-C. Chang, and G. Russello, "Scalable private decision tree evaluation with sublinear communication," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 843–857.
[32] P. Pullonen *et al.*, "Actively secure two-party computation: Efficient beaver triple generation," *Instructor*, 2013.
[33] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with honest majority," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, 2019, pp. 307–328.
[34] T. S. Daniel Demmler and M. Zohner. A framework for efficient mixed-protocol secure two-party computation. [Online]. Available: https://github.com/encryptogroup/ABY
[35] J. Doerner and abhi shelat. Jack doerner and abhi shelat. [Online]. Available: https://gitlab.com/neucrypt/floram
[36] Q. L. Yuheng Xia and J. F. et.al, accessed: Apr 23, 2024. [Online]. Available: https://github.com/XidianNSS/NssMPClib
[37] D. G. Aldrich. Uci machine learning repository: Data sets. [Online]. Available: https://archive.ics.uci.edu/ml/index.php
[38] J. Doerner and A. Shelat, "Scaling oram for secure computation," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 523–535.