UC Secure Private Branching Program and Decision Tree Evaluation

Keyu Ji, Bingsheng Zhang, Tianpei Lu, Lichun Li, and Kui Ren *

Abstract

Branching program (BP) is a DAG-based non-uniform computational model for L/poly class. It has been widely used in formal verification, logic synthesis, and data analysis. As a special BP, a decision tree is a popular machine learning classifier for its effectiveness and simplicity. In this work, we propose a UC-secure efficient 3-party computation platform for outsourced branching program and/or decision tree evaluation. We construct a constant-round protocol and a linear-round protocol. In particular, the overall (online + offline) communication cost of our linear-round protocol is $O(d(\ell + \log m + \log n))$ and its round complexity is $2d - 1$, where m is the DAG size, n is the number of features, ℓ is the feature length, and d is the longest path length. To enable efficient oblivious hopping among the DAG nodes, we propose a lightweight 1-out-of-N shared OT protocol with logarithmic communication in both online and offline phase. This partial result may be of independent interest to some other cryptographic protocols. Our benchmark shows, compared with the state-of-the-arts, the proposed constant-round protocol is up to 10X faster in the WAN setting, while the proposed linear-round protocol is up to 15X faster in the LAN setting.

1 Introduction

Branching program (BP) or binary decision diagram is a nonuniform computational model for L/poly class. The computation is specified by a directed acyclic graph (DAG) with a unique source node and several sink nodes; an evaluation is usually performed by traverse from the source node to a sink node. BP has been widely used in formal verification, logic synthesis and data analysis, etc. In particular, decision tree is a special case of BP, known for its effectiveness and simplicity as a machine learning classifier with a number of useful applications, including credit-risk assessment, spam classification, medical diagnosis.

Privacy concerns often raise, when sensitive information are involved. In the past decades, the privacy-preserving BP and decision tree evaluation problem has been extensive studied in the literature [1–9]. These works can be divided into two main categories based on protocol round complexity: (i) constant-round solutions [1, 2, 4–6], and (ii) linear-round solutions [8–11] whose round complexity is linear in the longest path length d. As summarized in [7], a typical constant-round solution consists of three functional modules: (a) private feature selection, (b) secure comparison, and (c) oblivious path evaluation. Each step can be realized by either garbled circuit or homomorphic encryption based protocols. The overall protocol usually needs to obliviously evaluate each decision node of the DAG for privacy preservation; therefore, they are suitable for BPs and decision trees with small DAG size, say less than 2^{20}. On the other hand, linear-round solutions can bypass this limitation by obliviously hopping along a DAG path according to the outcome of previous decision nodes. This is known as *oblivious access index* (OAI) [9], which can be realized by either OT or ORAM. The OT-based OAI private decision tree evaluation protocol proposed in [9] takes linear communication (in tree size, m) and $4d$ rounds. When OAI is realized by Circuit ORAM [12], the online communication complexity can be reduced to $O(d^4)$, but it takes up to $O(d^2)$ rounds.

The best linear-round solution is recently proposed by Ma *et al.* [11]. It reduces the online communication cost to $O(d)$ using key management and conditional OT. However, prior to each evaluation, the model owner has to prepare and share a one-time encoding of the tree to the client, which leads to

*K. Ji, B. Zhang, T. Lu and K. Ren are with College of Computer Science and Technology, Zhejiang University, Hangzhou, China. E-mail: {jikeyu, bingsheng, lutianpei, kuiren}@zju.edu.cn. L. Li is with Ant Group, Hangzhou, China. E-mail: lichun.llc@antgroup.com. B. Zhang is Corresponding Author.

Table 1: Performance comparison: m is the DAG(or decision tree) size, m_c is the number of decision nodes, $\widetilde{m}$ is the DAG(or decision tree) size after depth-padding, $\widetilde{m}_c$ is the number of decision nodes in padded tree, n is the number of features, N is the number of model owners, ℓ is the bit-length of feature and classification value, λ_1 is the size of symmetric ciphertext (= 128), λ_2 is the size of ElGamal ciphertext (= 514), λ_3 is the size of DGK ciphertext (= 2048), λ_4 is the size of Paillier ciphertext (= 4096), λ_5 is the size of BGV(SWHE) public key, λ_6 is the size of BGV ciphertext, λ_7 is the size of MKBGV ciphertext, λ_8 is the size of AES key (= 128).

Scheme	Communication		Rounds	Outsourcing
	offline	online		
[3] (GGG)	$((n + \widetilde{m}_c)\log n + 2m_c \log \widetilde{m}_c - n + 2 + 2\widetilde{m}_c)\ell\lambda_1 + 2\widetilde{m}_c(\lambda_1 + \log(\widetilde{m}_c + 1))$	$n\ell(2\lambda_1 + 1)$	2	○
[6] (HHH)	-	$((m_c + n)\ell + 2(m_c + 1) + n)\lambda_2$	4	○
[7] (GGH)	$((n + m_c)\log n + 2m_c \log m_c - n + 2 + 2m_c)\ell\lambda_1$	$n\ell(2\lambda_1 + 1) + (3m_c + 2)\lambda_2$	4	○
[7] (HGH)	$5m_c\ell\lambda_1$	$(m_c + n)\lambda_3 + m_c\ell(2\lambda_1 + 1) + (3m_c + 2)\lambda_2$	6	○
[13]**	-	$n\ell\lambda_1 + \lambda_5 + \lambda_6 + (2N + 5)\lambda_7$	2	●
Ours (const-round)	$2^d(3\log n + \ell)\lambda_8$	$2^d(6\log n + 4\ell + \lambda_8) + 2d\lambda_8$	4	●
[10]	$6(2^d n\ell + d(3\ell - \log l - 2) + 2^d - 1)$	$4(2^d n\ell + d(3\ell - \log l - 2) + 2^d - 1)$	$\log \ell + d + 1$	●
[14]	$6((2^d n + 4)\ell - 5)$	$4((2^d n + 4)\ell - 5)$	$2\ell - 1$	●
[15]**	$6(2^d - 1)\ell$	$3 \cdot 2^{d-1}\lambda_4 + 4(2^d - 1)\ell$	$d + 1$	●
[9] (OT)	$6d\ell\lambda_1$	$d((m + n)\ell + 2(\log m + \log n)\lambda_1)$	$4d$	○
[11] (complete)	$2^d(\ell + \log n)$	$d(4\lambda_1 + n\ell + (7\ell + 8)\lambda_1)$	$2d - 1$	●
[11] (sparse)	$m(\ell + \log n + \lambda_1 + 3d)$	$d((4\lambda_1 + n\ell) + (7\ell + 8)\lambda_1 + 8)$	$2d - 1$	○
Ours (linear-round)	$12d(\log n + \log m + \ell)\lambda_8$	$12d(3(\log m + \log n) + 2\ell)$	$2d - 1$	●

** Those protocols do not hide the feature index from the servers.

linear communication in the offline phase. Meanwhile, the protocol proposed by [11] can be modified to fit the outsourcing setting, where the model owner and the data owner just need to share their private input to the computing servers without heavily involved in the evaluation process. This setting enables the usage scenarios when the features are spited among multiple clients, and it is friendly to mobile devices with low-computation resources, such as IoT sensors. However, their outsourcing solution [11] needs to pad the decision tree to a complete tree for privacy preservation, and it costs $O(2^d)$ communication to refresh the shared decision tree in the offline phase of each evaluation. In addition, their solution does not naturally support BP evaluation.

1.1 Our approach

In this work, we investigate the outsourced private branching program and decision tree evaluation problem. Our approach follows the line of research initiated by Boyle *et al.* [16], which introduces the *distributed point function* (DPF). DPF enables an efficient two-server PIR protocol, where two servers hold the same set of messages $\mathbf{x}$, and the client wants to obliviously fetch x_i. Namely, the client first generates a pair of DPF keys encoding a point function $f_i(x)$, which has only one non-zero output, 1, when the input is i. The client then distributes the DPF keys to the two servers, and the servers jointly evaluate and return $x_i := \sum_{j=0}^{N-1} f_i(j) \cdot x_j$. Later, Doerner *et al.* [17] adopt DPF in the MPC setting to achieve ORAM. In [17], both servers S_0 and S_1 hold encrypted messages $\widetilde{x}_j := x_j \oplus \mathsf{PRF}_k(j)$, $j \in \mathbb{Z}_N$, where k is shared between them. For a given shared index $i \in \mathbb{Z}_N$, S_0 and S_1 first generates the DPF keys for $f_i(x)$ via MPC. After obtaining the shared $\widetilde{x}_i$, S_0 and S_1 then needs to obliviously evaluate $\mathsf{PRF}_k(i)$ via MPC to decrypt x_i. Therefore, the entire process is time-consuming. Recently, [18] introduce a 3-party DPF-based distributed read protocol with semi-honest security for a single corrupted party. It eliminates the needs of aforementioned two costly MPC operations by introducing replicated shares. However, the DPF keys [18] used are related to the secret input and also require a lot of communication.

We promote the 3-party distributed read protocol of [18] to an efficient shared OT protocol in online/offline model, where the costly computation and communication of the DPF key generation and distribution are transferred to the offline phase.

Our constant-round solution. We construct a 4-round private decision tree evaluation protocol, using the proposed 1-out-of-N shared OT protocol as a building block. We assume the model and features are already shared among the three servers. Note that the model needs to be padded to a complete tree to avoid privacy leakage. In the first round, the servers obliviously select corresponding features for all decision nodes. In the second round, for each decision node, a secure comparison is performed using *distributed interval containment function* (DICF) [19]. More specifically, S_2 plays the role of DICF key generator while S_0 and S_1 play the role of DICF evaluators. In the offline phase, S_2 precomputes the DICF keys and distribute them to S_0 and S_1. In the online phase, the servers mask the difference of its threshold and feature, and open it to S_0 and S_1. They then jointly evaluate DICF to securely compare the corresponding feature with the threshold. When the feature is less than the threshold, S_0 and S_1 obliviously set the left out-going edge cost of the decision node to 0 and the right out-going edge cost to a random value; vice versa. In the third round, for each leaf node of the decision tree, S_0 and S_1 sum up the edge costs along the path to get its path cost. They then cyclic shift the vector of path costs of all the leaf nodes together with the corresponding classification values, and jointly generate a random vector to mask the shifted classification values. After that, S_0 and S_1 open the shifted path costs and masked shifted classification values to S_2. In the fourth round, S_2 generates a pair of DPF keys according to the location of path cost is 0, and distributes keys to S_0 and S_1. Finally, S_2 outputs the masked classification value of the leaf node whose path cost is 0 to the receiver, while S_0 and S_1 can output the corresponding mask in the shared form.

Our linear-round solution. For large decision trees (and BP DAGs), we construct a $2d$-round private decision tree and BP evaluation protocol as follows. Our protocol supports sparse trees, and it only needs to pad one dummy node instead of transforming the model into a complete tree. The dummy node points to itself and all sink nodes point to it. For uniformity, besides sink nodes, all the other nodes have a dummy classification value 0. The protocol takes d steps with 2 rounds each. For each step along the evaluation path, the servers first invoke the shared OT protocol to obliviously fetch the current node together with its corresponding feature; they then jointly perform a *conditional shared OT* (CSOT) to determine the index of the next node together with the corresponding feature index. In a CSOT, the servers want to obliviously obtain one of two (shared) messages in the shared form based on a secure comparison result. It can be realized by a DICF evaluation and then a shared multiplication, but it would take 2 rounds. To reduce round complexity, we divide the four servers into two groups. Each group independently evaluates a DICF to perform secure comparison between the corresponding threshold and feature in parallel. Subsequently, the shared multiplication can be reduced to a scalar product which can be evaluated locally without further communication. Once a sink node is reached, the servers would obliviously evaluate the dummy node (repeatedly) until the protocol reaches d total steps. The classification values of all nodes in the evaluation path are summed to the final result.

Performance. Table 1 shows the communication and round complexity comparison between our scheme and the related works. The schemes that supports outsourcing are marked with $\bullet$. m is the DAG size, m_c is the number of decision nodes, $\widetilde{m}$ is the DAG size after depth-padding, $\widetilde{m}_c$ is the number of decision nodes in padded tree, n is the number of features, ℓ is the bit-length of feature and classification value. We emphasize that the concrete security parameters vary a lot among different schemes, and we use $\lambda_1, \ldots, \lambda_8$ to differentiate them. For instance, λ_4 refers to the ciphertext size of Paillier encryption, which is 4096 bits; whereas, the security parameter λ_8 is the 128-bit AES key size in our schemes. Note that some works (marked with **), e.g., [13, 15] do not protect the feature indices from the servers.

Our constant-round protocol supports outsourcing without the leakage of feature index, but it needs to pad the DAG to a complete tree; therefore, its communication size linearly depends on 2^d; yet it has the best performance for small tree evaluations in the WAN setting when the network delay is 80ms. (cf. Sec. 7) With regards to linear-round solutions, [11] is the most efficient scheme in the literature; nevertheless, their offline communication depends on the tree size, and complete tree padding is needed to support outsourcing. Our linear-round scheme has logarithmic communication in both online and offline phase.

2 Preliminaries

Notations. Throughout this paper, we use the following notations and terminologies. Let $\lambda \in \mathbb{Z}$ be the security parameter. Denote a value x indexed by a label b as $x^{(b)}$, while x^b means the value of x power of b. Denote a $(2,2)$-additive secret sharing in $\mathbb{Z}_n$ by $[\![x]\!] := \{x^{(0)}, x^{(1)}\}$, where $x^{(0)} + x^{(1)} = x \pmod n$ and S_j holds $x^{(j)}$ for $j \in \mathbb{Z}_2$. Denote a $(3,3)$-additive secret sharing in $\mathbb{Z}_n$ by $\langle x \rangle := \{x^{(0)}, x^{(1)}, x^{(2)}\}$, where S_j holds $x^{(j)}$ for $j \in \mathbb{Z}_3$, such that $x^{(0)} + x^{(1)} + x^{(2)} = x \pmod n$. Denote a $(3,2)$-additive sharing in $\mathbb{Z}_n$ by $\langle x \rangle^{\mathsf{rep}} := \{x^{(0)}, x^{(1)}, x^{(2)}\}$, where S_0 holds $\{x^{(0)}, x^{(1)}\}$, S_1 holds $\{x^{(1)}, x^{(2)}\}$, and S_2

holds $\{x^{(2)}, x^{(0)}\}$, such that $x^{(0)}+x^{(1)}+x^{(2)} = x \pmod n$. Namely, $\langle x\rangle^{\mathsf{rep}}$ is a replicated secret sharing. When K is a set, $k \leftarrow K$ stands for sampling k uniformly at random from K, and $|K|$ stands for the size of K in terms of the number of elements. When f is a algorithm, $y \leftarrow f(x)$ stands for running f on input x. We map $x \in \left[-2^{\ell-1}, 2^{\ell-1}\right]$ to $\mathbb{Z}_{2^\ell}$, i.e., when x is negative, $x' = x + 2^{\ell-1}$. Denote the decomposition point of positive and negative numbers as $\tau := 2^{\ell-1} - 1$.

Branching Program and Decision Tree. In this work, we focus on the deterministic branching program based on DAG and support its generalizations to integer-valued sink labels and input features. Let $\mathcal{B}$ denote a branching program. $\mathcal{B}$ has a unique source node and one or more sink nodes. Each non-sink node of $\mathcal{B}$ corresponds to an input feature $x \in \mathbb{Z}_{2^\ell}$ and has two outgoing edges labeled 0 or 1. Each sink node of $\mathcal{B}$ has a label $v_i \in \mathbb{Z}_{2^\ell}$ that determines the output of $\mathcal{B}$ evaluation. For a $\mathcal{B}$, m is defined as the number of its nodes, m_c is defined as the number of its non-sink nodes, and its depth d is the length of the longest path.

A decision tree is a special branching program whose underlying DAG is a tree. Denote a decision tree by $\mathcal{T}$. Without loss of generality, we assume $\mathcal{T}$ is a binary tree, which can be met by converting a general tree to a binary tree. $\mathcal{T}$ follows the notations of $\mathcal{B}$. The leaves and root in $\mathcal{T}$ correspond to the sinks and source node in $\mathcal{B}$, respectively. In addition, each non-sink node of $\mathcal{T}$ has a comparison function for input feature $x \in \mathbb{Z}_{2^\ell}$ and a given threshold $t \in \mathbb{Z}_{2^\ell}$.

The evaluation of $\mathcal{T}$ or $\mathcal{B}$ is performed by traversing from the source node to a sink node. Thus the evaluation takes linear time with respect to d. In detail, $\mathcal{T}$ or $\mathcal{B}$ receives an n-dimensional feature vector $\mathbf{x} := (x_i)_{i\in\mathbb{Z}_n}$ as evaluation input. Starting from the source node, for the i-th node, if current node is a non-sink node, fetch x_{k_i} from $\mathbf{x}$, where $k_i \in \mathbb{Z}_n$ is the index of the corresponding feature. Then determine the next node as

$$c \leftarrow (x_{k_i} < t_i) \text{ for } \mathcal{T}, \text{ or}$$
$$c \leftarrow x_{k_i} \text{ for } \mathcal{B}.$$

If $c = 1$, the next node is connected to outgoing edge labeled 1 of the current node; otherwise, if $c = 0$, the next node is connected to outgoing edge labeled 0 of the current node. If current node is a leaf node (or sink node), the attached v_i is outputted as evaluation result. We refer to the path from the root to a leaf (or from the source node to the sink node) as the evaluation path for given $\mathbf{x}$.

In addition, we use depth-padding to indicate that dummy nodes are introduced in $\mathcal{B}$ or $\mathcal{T}$ such that its evaluation path for each input $\mathbf{x} \in (\mathbb{Z}_{2^\ell})^n$ has the same length. $\mathcal{B}'$ (or $\mathcal{T}'$) stands for $\mathcal{B}$ (or $\mathcal{T}$) after depth-padding, while $\widetilde{m}$ is defined as the number of its nodes and $\widetilde{m}_c$ is defined as the number of its non-sink nodes.

Function Secret Sharing. *Function Secret Sharing* (FSS) is introduced by Boyle *et al.* [16]. Given a function family $\mathcal{F} = \{f(x) : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}\}$, a dealer uses the FSS scheme for $\mathcal{F}$ to split a function $f(x) \in \mathcal{F}$ into two additive shares $[\![f(x)]\!] := \{f^{(0)}(x), f^{(1)}(x)\}$, such that $\forall x \in \mathbb{G}^{in}$, $f^{(0)}(x) + f^{(1)}(x) = f(x) \pmod{|\mathbb{G}^{\mathsf{out}}|}$.

Distributed Point Function (DPF) is an FSS scheme for the point function $f_{\alpha,\beta}(x) : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}$ whose range only has one non-zero value $f_{\alpha,\beta}(\alpha) = \beta$. It consists of algorithms Gen and Eval defined as follows:

- $\mathsf{Gen}(1^\lambda, f_{\alpha,\beta})$ is a key generation algorithm that outputs a pair of keys $(\mathcal{K}^{(0)}, \mathcal{K}^{(1)})$. Each key includes a random PRF seed s and $\lceil \log_2 |\mathbb{G}^{\mathsf{in}}| \rceil + 1$ correction words. Each key is able to efficiently describe the share of $f_{\alpha,\beta}$ without revealing α, β.
- $\mathsf{Eval}(b, \mathcal{K}^{(b)}, x)$ is an evaluation algorithm. $\forall x \in \mathbb{G}^{\mathsf{in}}$, $\forall b \in \mathbb{Z}_2$, it outputs $\beta_x^{(b)} \in \mathbb{G}^{\mathsf{out}}$, such that $\beta_x^{(0)} + \beta_x^{(1)} = f_{\alpha,\beta}(x) \pmod{|\mathbb{G}^{\mathsf{out}}|}$.

When DPF is used to realize a PIR protocol, the servers need to run Eval on every element of the input domain, named *full domain evaluation*. [20] provides a more efficient scheme for this case, rather than executing $|\mathbb{G}^{\mathsf{in}}|$ independent invocations of Eval. We adopt their scheme and denote it by $\mathsf{EvalAll}(b, \mathcal{K}^{(b)})$.

Distributed Comparison Function (DCF) is an FSS scheme for the comparison function $f^{<}_{\alpha,\beta}(x) : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}$, which outputs β if $0 \le x < \alpha$ and outputs 0 if $x \ge \alpha$. Based on the DCF scheme, [19] provides the *Distributed Interval Containment Function* (DICF) construction to compute interval containment for a secret input and a publicly known interval. Denote the interval containment function as $f^{\mathsf{IC}}_{p,q}(x) : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}$, which outputs 1 if $x \in [p, q]$ and outputs 0 otherwise. DICF is an FSS scheme for the *offset* interval containment function $f^{\mathsf{IC}}_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}(x) : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}$ with given random offset $r^{\mathsf{in}}, r^{\mathsf{out}}$,

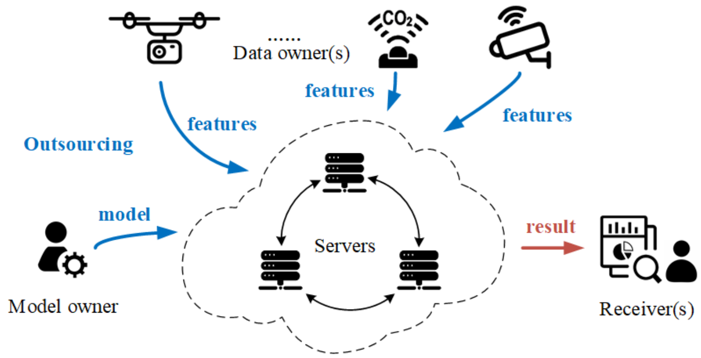


Figure 1: System architecture.

such that $f^{\mathsf{IC}}_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}(x+r^{\mathsf{in}})-r^{\mathsf{out}} = f^{\mathrm{IC}}_{p,q}(x)$. Similar to DPF, DICF also consists of a pair of algorithms $(\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ as follows:

- $\mathsf{Gen}^{\mathsf{IC}}_{p,q}(1^\lambda, f^{\mathsf{IC}}_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}})$ generates $(\mathcal{K}^{(0)}, \mathcal{K}^{(1)})$. Each key is able to efficiently describe the share of $f^{\mathsf{IC}}_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}$ with publicly known p, q but without revealing $r^{\mathsf{in}}, r^{\mathsf{out}}$.
- $\mathsf{Eval}^{\mathsf{IC}}_{p,q}(b, \mathcal{K}^{(b)}, x+r^{\mathsf{in}})$ outputs $\beta^{(b)}$ for $b \in \mathbb{Z}_2$, such that $\beta^{(0)} + \beta^{(1)} - r^{\mathsf{out}} = f^{\mathrm{IC}}_{p,q}(x) \pmod{|\mathbb{G}^{\mathsf{out}}|}$.

In the rest of paper, we focus on the case of $r^{\mathsf{out}} = 0$ and thus omit r^{out} in the offset interval containment function and the parameters of the DICF key generation algorithm.

Definition 1. *Let $T \subset [2]$. We say a two-party FSS scheme* $(\mathsf{Gen}, \mathsf{Eval})$ *is T-secure for function family $\mathcal{F} = \{f : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}\}$, if for all non-uniform PPT adversaries $\mathcal{A}$, it holds that*

$$\mathsf{Adv}(1^\lambda, \mathcal{A}) = \left| \Pr\left[\begin{array}{l} (f_1, f_2, \phi) \leftarrow \mathcal{A}(1^\lambda); b \leftarrow \{1,2\}; \\ (\mathcal{K}^{(0)}, \mathcal{K}^{(1)}) \leftarrow \mathsf{Gen}(1^\lambda, f_b); \\ b^* \leftarrow \mathcal{A}((\mathcal{K}^{(i)})_{i\in T}, \phi) : \\ f_1, f_2 \in \mathcal{F} \ \wedge \ b = b^* \end{array} \right] - \frac{1}{2} \right|$$

is negligible in λ.

3 System Architecture and Security Model

System Architecture. Fig. 1 gives a high-level architecture of our outsourced private decision tree and BP evaluation platform. The entities consists of a set of three non-colluding computing servers $\mathcal{S} := \{S_0, S_1, S_2\}$, the model owner M, the data owner D, and the receiver R. Initially, the model owner shares its model $\mathcal{M}$ among the computing servers. For each evaluation, a subset of data owners provide their feature data to the computing servers in the shared form; the servers then obliviously evaluate the model on given data and output the result to a subset of the receivers.

Universal Composability. Our security model is based on the Universal Composibility (UC) framework [21], which lays down a solid foundation for designing and analyzing protocols secure against attacks in an arbitrary *network* execution environment (therefore it is also known as *network aware security model*). Roughly speaking, in the UC framework, protocols are carried out over multiple interconnected machines; to capture attacks, a network adversary $\mathcal{A}$ is introduced, which is allowed to corrupt some machines (i.e., have the full control of all physical parts of some machines); in addition, $\mathcal{A}$ is allowed to partially control the communication tapes of all uncorrupted machines, that is, it sees all the messages sent from and to the uncorrupted machines and controls the sequence in which they are delivered. Then, a protocol ρ is a UC-secure implementation of a functionality $\mathcal{F}$, if it satisfies that for every network adversary $\mathcal{A}$ attacking an execution of ρ, there is another adversary $\mathcal{S}$—known as the simulator—attacking the ideal process that uses $\mathcal{F}$ (by corrupting the same set of machines), such that, the executions of ρ with $\mathcal{A}$ and that of $\mathcal{F}$ with $\mathcal{S}$ makes no difference to any network execution environment.

The idea world execution. In the ideal world, the computing servers $\mathcal{S} := \{S_0, \ldots, S_{\kappa-1}\}$, the model owner M, the data owner D, and the receiver R only communicate with an ideal functionality $\mathcal{F}^{\kappa}_{\mathsf{bp}}$

Functionality $\mathcal{F}_{\mathsf{bp}}^{\kappa}$

It interacts with the model owner M, the data owner D, the receiver R, a set of computing servers $\mathcal{S} := \{S_0, \ldots, S_{\kappa-1}\}$, and the adversary Sim. It is parameterized with a set $\mathcal{J}$ and a variable status.
Initially, set $\mathcal{J} := \emptyset$ and $\mathsf{status} := 0$.

Outsourcing phase:

- Upon receiving $(\textsc{Model}, \mathsf{sid}, \mathcal{M})$ from M:
  - Send notification $(\textsc{Model}, \mathsf{sid}, M, (\mathcal{M}.m, \mathcal{M}.d))$ to Sim;
  - Set $\mathsf{status} := 1$;
  - Record $\mathcal{M}$;
- Upon receiving $(\textsc{Data}, \mathsf{sid}, \mathbf{x})$ from D, if $\mathsf{status} = 1$:
  - Send notification $(\textsc{Data}, \mathsf{sid}, D, |\mathbf{x}|)$ to Sim;
  - Set $\mathsf{status} := 2$;
  - Record $\mathbf{x}$;

Evaluation phase:

- Upon receiving $(\textsc{Eval}, \mathsf{sid})$ from server $S_i \in \mathcal{S}$, if $\mathsf{status} = 2$ does:
  - Send notification $(\mathsf{Eval}, \mathsf{sid}, S_i)$ to Sim;
  - Set $\mathcal{J} := \mathcal{J} \cup \{S_i\}$;
  - If $|\mathcal{J}| = \kappa$, run $y \leftarrow \mathcal{M}(\mathbf{x})$;
  - Send $(\textsc{Result}, \mathsf{sid}, y)$ to R via private input delayed channel.

Figure 2: The ideal functionality $\mathcal{F}_{\mathsf{bp}}^{\kappa}$

Functionality $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$

It interacts with $\mathcal{S} := \{S_0, S_1, S_2\}$ and the adversary Sim.

- Upon receiving $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{x}^{(j+1 \pmod 3)}, i^{(j)})$ from $S_j \in \mathcal{S}$, where $\mathbf{x}^{(j)} := (x_0^{(j)}, \ldots, x_{N-1}^{(j)})$:
  - Send notification $(\textsc{Fetch}, \mathsf{sid}, S_j)$ to Sim;
  - Record $(\mathbf{x}^{(j)}, \mathbf{x}^{(j+1 \pmod 3)}, i^{(j)})$;
- Once all players have submitted their input, does:
  - Assert the same name shares are identical.
  - Compute $i := \sum_{j=0}^{2} i^{(j)} \pmod N$;
  - Upon receiving $(\textsc{Rand}, \mathsf{sid}, y^*)$ from Sim for the corrupted party S_k:
    * Pick random $y^{(0)}, y^{(1)}, y^{(2)} \in \mathbb{Z}_{2^\ell}$ s.t. $y^{(k)} = y^*$ and $\sum_{j=0}^{2} y^{(j)} = \sum_{j=0}^{2} x_i^{(j)} \pmod{2^\ell}$;
    * Send $(\textsc{Return}, \mathsf{sid}, y^{(j)})$ to all parties $S_j \in \mathcal{S}$ via private delayed channel.

Figure 3: The shared OT functionality $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$

during the execution. As depicted in Fig. 2, the ideal functionality $\mathcal{F}_{\mathsf{bp}}^{\kappa}$ consists of two phases. In the outsourcing phase, the model owner M sends its model $\mathcal{M}$ to the ideal functionality. Later, the data owner D sends its data $\mathbf{x}$ to the ideal functionality. Note that the size and depth of the model as well as the number of features are leaked to the adversary Sim. During the evaluation phase, once all computing servers has sent $(\textsc{Eval}, \mathsf{sid})$ to the functionality $\mathcal{F}_{\mathsf{bp}}^{\kappa}$, $\mathcal{F}_{\mathsf{bp}}^{\kappa}$ runs $y \leftarrow \mathcal{M}(\mathbf{x})$ and then sends $(\textsc{Result}, \mathsf{sid}, y)$ to R via input delayed channel.

The real world execution. In the real world, the model owner M, the data owner D, and the receiver R, only communicate with the computing servers $\mathcal{S} := \{S_0, \ldots, S_{\kappa-1}\}$ to submit the input and/or obtain the output. While the computing servers jointly evaluate the model with privacy preservation. The protocols are described in Sec. 6, below.

Definition 2. *We say protocol Π UC-secure realizes functionality $\mathcal{F}_{\mathsf{bp}}^{\kappa}$ if for all PPT adversaries $\mathcal{A}$ there exists a PPT simulator Sim such that for all PPT environment $\mathcal{Z}$ it holds:*

$$\mathsf{Exec}_{\Pi,\mathcal{A},\mathcal{Z}} \approx \mathsf{Exec}_{\mathcal{F}_{\mathsf{bp}}^{\kappa},\mathsf{Sim},\mathcal{Z}}$$

Protocol $\Pi_{\mathsf{sot}}^{N,\ell}$

Initialization:
- S_0 and S_1 agree on a random seed $\eta_0 \leftarrow \{0,1\}^\lambda$;
- S_1 and S_2 agree on a random seed $\eta_1 \leftarrow \{0,1\}^\lambda$;
- S_2 and S_0 agree on a random seed $\eta_2 \leftarrow \{0,1\}^\lambda$.

Offline phase:
- Upon initialization, $S_j, j \in \mathbb{Z}_3$ does:
  - Generate $\varphi_j \leftarrow \mathbb{Z}_N$;
  - Set $(\mathcal{K}_{\varphi_j}^{(0)}, \mathcal{K}_{\varphi_j}^{(1)}) \leftarrow \mathsf{DPF.Gen}(1^\lambda, f_{\varphi_j,1})$ for the point function $f_{\varphi_j,1} : \mathbb{Z}_N \to \mathbb{Z}_{2^\ell}$;
  - Send $(\mathsf{sid}, \mathcal{K}_{\varphi_j}^{(0)})$ to S_{j+1}, $(\mathsf{sid}, \mathcal{K}_{\varphi_j}^{(1)})$ to S_{j+2};

Online phase:
- Upon receiving $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{x}^{(j+1)}, i^{(j)})$ from the environment $\mathcal{Z}$, the player $S_j, j \in \mathbb{Z}_3$ does:
  - For $k \in \mathbb{Z}_3$, set:
    - $w_{k,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_N}(\mathsf{sid}, k)$, $w_{k,j+2} \leftarrow \mathsf{PRF}_{\eta_{j+2}}^{\mathbb{Z}_N}(\mathsf{sid}, k)$;
    - $\delta_k^{(j)} := i^{(j)} - w_{k,j} + w_{k,j+2} \pmod N$;
  - Set $\delta_j^{(j)} := \delta_j^{(j)} - \varphi_j \pmod N$;
  - Send $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+1}^{(j)})$ to S_{j+2}, $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+2}^{(j)})$ to S_{j+1};
- Upon receiving $(\mathsf{sid}, \delta_{j+1}^{(j+1)}, \delta_{j+2}^{(j+1)})$ from S_{j+1}, receiving $(\mathsf{sid}, \delta_{j+2}^{(j+2)}, \delta_{j+1}^{(j+2)})$ from S_{j+2}, the player $S_j, j \in \mathbb{Z}_3$ does:
  - For $k \in \mathbb{Z}_3/j$, set $\delta_k := \delta_k^{(1)} + \delta_k^{(2)} + \delta_k^{(3)} \pmod N$;
  - Set $(\beta_{k,\varphi_{j+1}}^{(1)})_{k \in \mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(1, \mathcal{K}_{\varphi_{j+1}}^{(1)})$;
  - Set $(\beta_{k,\varphi_{j+2}}^{(0)})_{k \in \mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(0, \mathcal{K}_{\varphi_{j+2}}^{(0)})$;
  - Set $y^{(j)} := \sum_{k=0}^{N-1} (x_{k+\delta_{j+1}}^{(j)} \cdot \beta_{k,\varphi_{j+1}}^{(1)} + x_{k+\delta_{j+2}}^{(j+1)} \cdot \beta_{k,\varphi_{j+2}}^{(0)})$;
  - Set $\zeta_j \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$, $\zeta_{j+2} \leftarrow \mathsf{PRF}_{\eta_{j+2}}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$;
  - Return $y^{(j)} := y^{(j)} + \zeta_j - \zeta_{j+2} \pmod{2^\ell}$.

Figure 4: Shared OT protocol $\Pi_{\mathsf{sot}}^{N,\ell}$. For simplicity, we omit $\pmod 3$ from the expressions with $j \in \mathbb{Z}_3$.

4 3-party Shared OT

In the shared OT, given shared data $\mathbf{x} := (x_0, \ldots, x_{N-1})$ and an shared index $i \in \mathbb{Z}_N$, the MPC players can jointly obtain x_i in the shared form without revealing i. We introduce an efficient 3-party shared OT protocol designed in the online/offline model. Analogy to [18], the data are replicated shared while the index is additively shared in our protocol. That is, S_0 holds $\{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, i^{(0)}\}$, S_1 holds $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, i^{(1)}\}$, and S_2 holds $\{\mathbf{x}^{(2)}, \mathbf{x}^{(0)}, i^{(2)}\}$, where $\mathbf{x} = \sum_{j=0}^{2} \mathbf{x}^{(j)}$ and $i = \sum_{j=0}^{2} i^{(j)} \pmod N$.

Intuition. Our construction is inspired by [18]. The main idea is that each MPC player serves as the generator of DPF scheme, when the other two players serve as evaluators to learn the i-th position value of their replicated data shares. More specifically, in the case of S_0 as the generator, [18] lets S_1 and S_2 first randomly pick $r^{(1)}, r^{(2)} \leftarrow \mathbb{Z}_N$ respectively, and then exchange $r^{(1)} - i^{(1)}$ and $r^{(2)} - i^{(2)}$ while sending $r^{(1)}, r^{(2)}$ to S_0. After that, the players S_1 and S_2 can compute $\delta := r^{(1)} - i^{(1)} + r^{(2)} - i^{(2)} \pmod N$; the player S_0 generates a pair of DPF keys for the point function $f_{\omega,1}(x)$, where $\omega := i^{(0)} + r^{(1)} + r^{(2)} \pmod N$, and distributes the DPF keys to S_1 and S_2. Finally, S_1 and S_2 full-domain evaluate the shared $[\![f_{\omega,1}(x)]\!]$ to correspondingly scale and sum the "shifted shares", which results by shifting the position of every entry in $\mathbf{x}^{(2)}$ by δ. Note that this leads to S_1 and S_2 holding $[\![x_i^{(2)}]\!]$. However, since ω is related to a share of the secret index i, [18] cannot pre-construct the shared $f_{\omega,1}(x)$ for reducing the online communication. To address this issue, we let S_0 generate and distribute DPF keys of $f_{\varphi,1}(x)$ in the offline phase, where $\varphi \leftarrow \mathbb{Z}_N$ is randomly selected by S_0. In the online phase, three players jointly compute and open $\langle\delta\rangle := \langle i\rangle + \langle 0\rangle - \varphi \pmod N$ to S_1, S_2. Subsequently, S_1 and S_2 can play the DPF evaluators to obtain the shared $x_i^{(2)}$ in the same way as [18].

Theorem 1. *Let* $\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}$ *be a secure function secret sharing scheme for point function* $f_{\alpha,\beta}(x) : \mathbb{Z}_N \mapsto \mathbb{Z}_{2^\ell}$ *with adversarial advantage* $\mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$*. Let* $\mathsf{PRF}^{\mathbb{Z}_N} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_N$ *be*

Functionality $\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1}$

It interacts with $\mathcal{S} := \{S_0, S_1, S_2\}$ and the adversary Sim.

- Upon receiving $(\textsc{Sel}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{m}^{(j)})$ from $S_j \in \mathcal{S}$:
  - Send notification $(\textsc{Sel}, \mathsf{sid}, S_j)$ to Sim;
  - Record $(\mathbf{x}^{(j)}, \mathbf{m}^{(j)})$;
- Once all players have submitted their input, does:
  - For $k \in \{0,1\}$, compute $x_k := \sum_{j=1}^{3} x_k^{(j)} \pmod{2^{\ell_0}}$ and $m_k := \sum_{j=0}^{2} m_k^{(j)} \pmod{2^{\ell_1}}$;
  - Set $b \leftarrow (m_0 < m_1)$;
  - Upon receiving $(\textsc{Rand}, \mathsf{sid}, y^*)$ from Sim for the corrupted party S_k:
    * Pick random $y^{(0)}, y^{(1)}, y^{(2)} \in \mathbb{Z}_{2^{\ell_0}}$ s.t. $y^{(k)} = y^*$ and $\sum_{j=0}^{2} y^{(j)} = x_{1-b} \pmod{2^{\ell_0}}$;
    * Send $(\textsc{Return}, \mathsf{sid}, y^{(j)})$ to all parties $S_j \in \mathcal{S}$ via private delayed channel.

Figure 5: The conditional shared OT functionality $\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1}$

a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A})$*. The protocol* $\Pi_{\mathsf{sot}}^{N,\ell}$ *as described in Fig. 4 UC-realizes* $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$ *as described in Fig. 3 against semi-honest adversaries who can statically corrupted up to* 1 *server with distinguishing advantage*

$$9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A}) + 2 \cdot \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \ .$$

Protocol description. Our 3-party shared OT protocol is symmetric as depicted in Fig. 4. To reduce the protocol communication, we assume that S_0 and S_1 agree on a random seed $\eta_0 \in \{0,1\}^\lambda$; S_1 and S_2 agree on a random seed $\eta_1 \in \{0,1\}^\lambda$; S_2 and S_0 agree on a random seed $\eta_2 \in \{0,1\}^\lambda$. We take these shared seeds as PRF keys to enable non-interactive random-number sharing using PRF. This method is, known as *Pseudorandom Secret Sharing* (PRSS), originally introduced in [22]. In the offline phase, S_0 generates a pair of DPF keys $(\mathcal{K}_{\varphi_0}^{(0)}, \mathcal{K}_{\varphi_0}^{(1)})$ for $f_{\varphi_0,1} : \mathbb{Z}_N \to \mathbb{Z}_{2^\ell}$, where $\varphi_0 \leftarrow \mathbb{Z}_N$ is randomly picked. S_0 then sends $\mathcal{K}_{\varphi_0}^{(0)}$ to S_1, $\mathcal{K}_{\varphi_0}^{(1)}$ to S_2. Repeat the above with S_1 and S_2 as the DPF generator. This results in S_1, S_2 sharing the point function $f_{\varphi_0,1}$, S_0, S_2 sharing $f_{\varphi_1,1}$, and S_0, S_1 sharing $f_{\varphi_2,1}$. In the online phase, the MPC players first construct several $\langle 0 \rangle$ using the random seeds and pseudorandom function PRF without communication. Next, for $j \in \mathbb{Z}_3$, three players jointly compute and reveal $\langle \delta_j \rangle := \langle i \rangle + \langle 0 \rangle - \varphi_j \pmod{N}$ to $S_{j+1 \pmod 3}$, $S_{j+2 \pmod 3}$. After that, S_1 and S_2 full-domain evaluate $\mathcal{K}_{\varphi_0}^{(0)}, \mathcal{K}_{\varphi_0}^{(1)}$ respectively to obtain a shared unit vector $(\llbracket \beta_{k,\varphi_0} \rrbracket)_{k \in \mathbb{Z}_N}$, and then jointly compute

$$\llbracket x_i^{(2)} \rrbracket := \sum_{k=0}^{N-1} (x_{k+\delta_0}^{(2)} \cdot \llbracket \beta_{k,\varphi_0} \rrbracket) \pmod{2^\ell}.$$

Similarly for obtaining $\llbracket x_i^{(0)} \rrbracket$ and $\llbracket x_i^{(1)} \rrbracket$. Note that $x_i = \sum_{j=0}^{2} x_i^{(j)} \pmod{2^\ell}$. Finally, we re-randomize shares to ensure their uniform distribution.

Efficiency. $\Pi_{\mathsf{sot}}^{N,\ell}$ is a one-round shared OT protocol with offline communication cost $6\lambda \log N$ bits and online communication cost 12ℓ bits.

Security. We show the security of our 1-out-of-N shared OT Protocol $\Pi_{\mathsf{sot}}^{N,\ell}$ with the following theorem, and its proof can be found in section A of the supplemental material.

5 Conditional Shared OT

In the conditional shared OT protocol, given a vector of shared messages $\mathbf{x} := (x_0, x_1) \in (\mathbb{Z}_{2^{\ell_0}})^2$ and two shared keywords $\mathbf{m} := (m_0, m_1) \in (\mathbb{Z}_{2^{\ell_1}})^2$, the MPC players first securely compare $b \leftarrow (m_0 < m_1)$ and then obtain x_{1-b} in the shared form without revealing b. As depicted in Fig. 5, our conditional shared OT is a 3-party computation protocol. The messages and keywords are additively shared among the 3 parties. Let $\mathbf{x}^{(j)} := (x_0^{(j)}, x_1^{(j)})$ and $\mathbf{m}^{(j)} := (m_0^{(j)}, m_1^{(j)})$ be the shares of player S_j, $j \in \mathbb{Z}_3$. We have $\mathbf{x} = \sum_{j=0}^{2} \mathbf{x}^{(j)}$ and $\mathbf{m} = \sum_{j=0}^{2} \mathbf{m}^{(j)}$.

Intuition. Naively, the conditional shared OT protocol can be realized by a secure comparison followed by a oblivious selection (a.k.a. multiplication) protocol. However, this would result a 2-round

Protocol $\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}$

Initialization:

- S_0 and S_1 agree on a random seed $\eta_0 \leftarrow \{0,1\}^\lambda$;
- S_1 and S_2 agree on a random seed $\eta_1 \leftarrow \{0,1\}^\lambda$;
- S_2 and S_0 agree on a random seed $\eta_2 \leftarrow \{0,1\}^\lambda$.

Offline phase:

- Upon initialization, $S_j, j \in \mathbb{Z}_3$ does:
  - Set $\rho_j \leftarrow \mathbb{Z}_{2^{\ell_1}}$, $\tau := 2^{\ell_1 - 1} - 1$;
  - Set $(\mathcal{K}_{\rho_j}^{(0)}, \mathcal{K}_{\rho_j}^{(1)}) \leftarrow \mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\rho_j,0}^{\mathrm{IC}})$ for the function $f_{0,\tau,\rho_j,0}^{\mathrm{IC}} : \mathbb{Z}_{2^{\ell_1}} \to \mathbb{Z}_{2^{\ell_0}}$;
  - Send $(\mathsf{sid}, \mathcal{K}_{\rho_j}^{(0)})$ to S_{j+1}, $(\mathsf{sid}, \mathcal{K}_{\rho_j}^{(1)})$ to S_{j+2}.

Online phase:

- Upon receiving $(\textsc{Sel}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{m}^{(j)})$ from the environment $\mathcal{Z}$, player $S_j, j \in \mathbb{Z}_3$ does:
  - For $k \in \mathbb{Z}_3$, set
    - $w_{k,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid}, k), w_{k,j+2} \leftarrow \mathsf{PRF}_{\eta_{j+2}}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid}, k)$;
    - $\delta_k^{(j)} := m_1^{(j)} - m_0^{(j)} - w_{k,j} + w_{k,j+2} \pmod{2^{\ell_1}}$;
  - Set $\delta_j^{(j)} := \delta_j^{(j)} + \rho_j \pmod{2^{\ell_1}}$;
  - For $i \in \mathbb{Z}_2$, set
    - $\zeta_{j,i} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_0}}}(\mathsf{sid}, i)$, $\zeta_{j+2,i} \leftarrow \mathsf{PRF}_{\eta_{j+2}}^{\mathbb{Z}_{2^{\ell_0}}}(\mathsf{sid}, i)$;
    - $\tilde{x}_i^{(j)} := x_i^{(j)} + \zeta_{j,i} - \zeta_{j+2,i} \pmod{2^{\ell_0}}$
  - Send $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+1}^{(j)}, \tilde{\mathbf{x}}^{(j)})$ to S_{j+2};
  - Send $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+2}^{(j)})$ to S_{j+1};
- Upon receiving $(\mathsf{sid}, \delta_{j+1}^{(j+1)}, \delta_{j+2}^{(j+1)}, \tilde{\mathbf{x}}^{(j+1)})$ from S_{j+1}, $(\mathsf{sid}, \delta_{j+2}^{(j+2)}, \delta_{j+1}^{(j+2)})$ from S_{j+2}, the player $S_j, j \in \mathbb{Z}_3$ does:
  - For $k \in \mathbb{Z}_3/j$, set $\delta_k := \delta_k^{(1)} + \delta_k^{(2)} + \delta_k^{(3)} \pmod{2^{\ell_1}}$;
  - Set $\beta_{j+1}^{(1)} \leftarrow \mathsf{DICF.Eval}_{0,\tau}^{\mathsf{IC}}(1, \mathcal{K}_{\rho_{j+1}}^{(1)}, \delta_{j+1})$;
  - Set $\beta_{j+2}^{(0)} \leftarrow \mathsf{DICF.Eval}_{0,\tau}^{\mathsf{IC}}(0, \mathcal{K}_{\rho_{j+2}}^{(0)}, \delta_{j+2})$;
  - Set $y^{(j)} := \tilde{x}_1^{(j)} + \sum_{q=0}^{1} \sum_{k=0}^{1} (-1)^k \cdot \tilde{x}_k^{(j+q)} \cdot \beta_{j+1+q}^{(1-q)}$;
  - Set $\zeta_{j,2} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_0}}}(\mathsf{sid}, 2)$, $\zeta_{j+2,2} \leftarrow \mathsf{PRF}_{\eta_{j+2}}^{\mathbb{Z}_{2^{\ell_0}}}(\mathsf{sid}, 2)$;
  - Return $y^{(j)} := y^{(j)} + \zeta_{j,2} - \zeta_{j+2,2} \pmod{2^{\ell_0}}$.

Figure 6: Conditional shared OT Protocol $\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}$. For simplicity, we omit $\pmod 3$ from the expressions with $j \in \mathbb{Z}_3$.

protocol. We compress the round complexity to one. In our protocol, any two servers form a group, i.e., there are three groups namely $\{S_0, S_1\}$, $\{S_1, S_2\}$ and $\{S_2, S_0\}$. Three players jointly compute and open $\delta := m_1 - m_0$ to each group with the corresponding DICF random offset. At the same round, we convert the shared massages $\langle \mathbf{m} \rangle$ from $(3,3)$-addictive secret sharing to replicated secret sharing. After that, each group holds a same share of message, and shares an offset interval containment function for δ. Therefore, the oblivious selection can be computed locally by scalar product of the replicated shares and the DICF evaluation result.

Protocol description. Our 1-round conditional shared OT is depicted in Fig. 6. During the initialization, S_0 and S_1 agree on a random seed $\eta_0 \in \{0,1\}^\lambda$; S_1 and S_2 agree on a random seed $\eta_1 \in \{0,1\}^\lambda$; S_2 and S_0 agree on a random seed $\eta_2 \in \{0,1\}^\lambda$. In the offline phase, for $j \in \mathbb{Z}_3$, the MPC player S_j generates DICF keys of the offset interval containment function $f_{0,\tau,\rho_j,0}^{\mathrm{IC}}(x) : \mathbb{Z}_{2^{\ell_1}} \to \mathbb{Z}_{2^{\ell_0}}$, where $\rho_j \in \mathbb{Z}_{2^{\ell_1}}$ is randomly picked. S_j then distributes the DICF keys to the servers S_{j+1} and S_{j+2}. In the online phase, three players jointly compute $\langle \delta_j \rangle := \langle m_1 \rangle - \langle m_0 \rangle + \langle 0 \rangle + \rho_j$ for $j \in \mathbb{Z}_3$, where $\langle 0 \rangle$ is constructed by random seeds and PRF, and then open $\langle \delta_j \rangle$ to S_{j+1}, S_{j+2}. Meanwhile, to build the replicated secret sharing of messages $\mathbf{x} := (x_0, x_1)$, S_0 sends re-randomized $(\tilde{x}_0^{(0)}, \tilde{x}_1^{(0)})$ to S_2, S_1 sends $(\tilde{x}_0^{(1)}, \tilde{x}_1^{(1)})$ to S_0, S_2 sends $(\tilde{x}_0^{(2)}, \tilde{x}_1^{(2)})$ to S_1. After that, servers evaluate DICF with the received keys and the masked δ to obtain the shared comparison result $[\![\beta_0]\!], [\![\beta_1]\!], [\![\beta_2]\!]$. Finally, servers locally

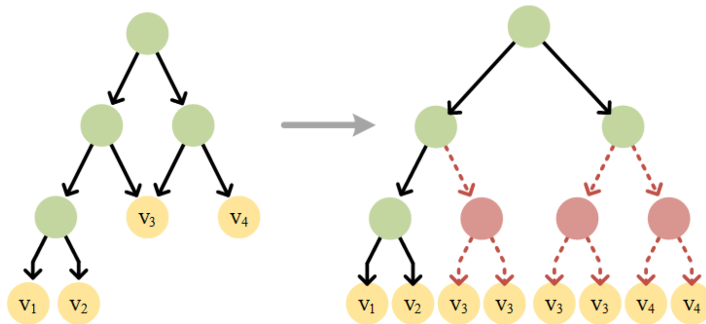


Figure 7: Complete tree depth-padding.

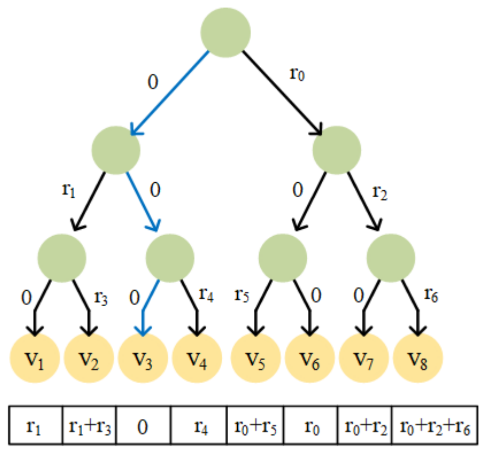


Figure 8: Path cost diagram.

compute scalar multiplication to get $\langle x_{1-b} \rangle$ as

$$\langle x_{1-b} \rangle := \sum_{j=0}^{2} (\llbracket \beta_j \rrbracket \cdot \tilde{x}_0^{(j+2)} + (1 - \llbracket \beta_j \rrbracket) \cdot \tilde{x}_1^{(j+2)}) \pmod{2_0^{\ell}}$$

and re-randomize the result to ensure the shares of the selected message in uniform distribution.

Efficiency. $\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}$ is a one-round protocol with offline communication cost $6\lambda\ell_1$ bits and online communication cost $6\ell_0 + 12\ell_1$ bits.

Security. We show the security of our conditional shared OT Protocol $\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}$ with the following theorem, and its proof can be found in section B of the supplemental material.

Theorem 2. *Let* $\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}$ *be a secure function secret sharing scheme for the offset interval containment function* $f^{\mathsf{IC}}_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}(x) : \mathbb{Z}_{2^{\ell_1}} \mapsto \mathbb{Z}_{2^{\ell_0}}$ *with adversarial advantage* $\mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$*. Let* $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_0}}$ *be a secure pseudorandom function with adversarial advantage* $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$*. Let* $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ *be a secure pseudorandom function with adversarial advantage* $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$*. The protocol* $\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}$ *as described in Fig. 6 UC-realizes* $\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1}$ *as described in Fig. 5 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage*

$$6 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) + 9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 2 \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) \ .$$

6 Private Decision Tree and BP Evaluation

In this section, we propose two solutions for outsourced private decision tree and BP evaluation. The first solution is a constant-round protocol for (small) complete trees; whereas, the second solution is a

Outsourcing Protocol $\Pi_{\mathsf{os}}^{\mathsf{const}}$

- Upon receiving ($\textsc{Model}, \mathsf{sid}, (\mathcal{P}, \mathbf{v})$) from the environment $\mathcal{Z}$, the model owner M:
  - **Foreach** element i in $\mathcal{P}$:
    - Set $k_i^{(0)} \leftarrow \mathbb{Z}_n$, $k_i^{(1)} := k_i - k_i^{(0)} \pmod n$
    - Set $t_i^{(0)} \leftarrow \mathbb{Z}_{2^\ell}$, $t_i^{(1)} := t_i - t_i^{(0)} \pmod{2^\ell}$;
    - Set $P_i^{(0)} := \{k_i^{(0)}, t_i^{(0)}\}$, $P_i^{(1)} := \{k_i^{(1)}, t_i^{(1)}\}$;
  - **Foreach** element i in $\mathbf{v}$:
    - $v_i^{(0)} \leftarrow \mathbb{Z}_{2^\ell}$, $v_i^{(1)} := v_i - v_i^{(0)} \pmod{2^\ell}$;
  - Send $(\mathcal{P}^{(0)}, \mathbf{v}^{(0)})$ to S_0, $(\mathcal{P}^{(1)}, \mathbf{v}^{(1)})$ to S_1.
- Upon receiving ($\textsc{Data}, \mathsf{sid}, \mathbf{x}$) from the environment $\mathcal{Z}$, the data owner D:
  - **Foreach** feature $x_i \in \mathbf{x}$:
    - Generate $x_i^{(0)}, x_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - Set $x_i^{(2)} := x_i - x_i^{(0)} - x_i^{(1)} \pmod{2^\ell}$;
  - **For** $j \in \mathbb{Z}_3$, send $(\mathbf{x}^{(j)}, \mathbf{x}^{(j+1 \pmod 3)})$ to S_j.

Figure 9: Outsourcing Protocol $\Pi_{\mathsf{os}}^{\mathsf{const}}$.

linear-round protocol for BP and (large) sparse tree evaluation.

6.1 Constant-Round Protocol

Our constant-round protocol requires four communication rounds and a complete decision tree, which can be transformed from a normal DAG by adding dummy nodes as illustrated in Fig. 7, i.e. $\widetilde{m} = 2^d - 1$, $\widetilde{m}_c = 2^{(d-1)} - 1$. All leaf nodes extended by dummy decision nodes have the same classification value as real path. We use a vector, denoted as $\mathcal{P}$, to represent all decision nodes and complete tree structure. Each $P_i \in \mathcal{P}$ consists of the input selection index k_i and a threshold value t_i. The left and right child of P_i are P_{2i+1} and P_{2i+2}, respectively. The leaf nodes' classification values form the other vector, denoted as $\mathbf{v}$.

Our protocol selects corresponding features and compares thresholds with them for all decision nodes. For each $P_i \in \mathcal{P}$, S_0 and S_1 obliviously set its "selected" out-going edge cost (based on the comparison result) to 0, and set the other out-going edge cost to random value. Then S_0 and S_1 sum up the share of edge costs along all paths to get a vector of path costs in a shared form. As shown in Fig 8, only one path cost takes the value of 0 and the corresponding leaf nodes' classification value is the evaluation result.

Outsourcing. First of all, the model owner M invokes $\Pi_{\mathsf{os}}^{\mathsf{const}}$ as described in Fig. 9 to generate the additive share of $\mathcal{P}, \mathbf{v}$ and distribute them to S_0 and S_1. This step only needs to be performed once for a given model. Before the start of each evaluation, the data owner D shares the input features $\mathbf{x} := (x_i)_{i \in \mathbb{Z}_n}$ to three servers in replicated secret sharing. After the exection, S_0 holds $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}$, S_1 holds $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}$, and S_2 holds $\mathbf{x}^{(2)}, \mathbf{x}^{(0)}$, such that $\mathbf{x}^{(j)} := (x_i^{(j)})_{i \in \mathbb{Z}_n}$ for $j \in \mathbb{Z}_3$ and $\mathbf{x} = \mathbf{x}^{(1)} + \mathbf{x}^{(2)} + \mathbf{x}^{(3)}$.

Evaluation. Our constant-round protocol follows the modular design framework of [7]. As depicted in Fig. 10, it consists of feature selection, comparison and path evaluation.

Feature selection. For each $P_i \in \mathcal{P}$, with the secret shared index $[\![k_i]\!]$ in S_0 and S_1, we construct $\langle k_i \rangle$ as follows: S_2 first sets $k_i^{(2)} := 0$, three servers then randomize their shares to build $(3, 3)$-secret-sharing of k_i. After that, three servers invoke our shared OT protocol described in Sec. 4 to obtain corresponding feature $\langle x_{k_i} \rangle$.

Comparison. Our comparison protocol is based on the DICF scheme [19], where S_2 plays the role of generator and S_0, S_1 play the role of evaluators. For each $P_i \in \mathcal{P}$, to avoid leaking features and thresholds to servers, in the offline phase, we let S_2 precompute a pair of DICF keys, which is used to compare the corresponding input value with a random value ρ_i. S_2 then distributes the keys to the DICF evaluators S_0 and S_1. In the online phase, MPC players jointly compute and open $\Delta x_i := t_i - x_{k_i} + \rho_i$ to S_0 and S_1. After that, S_0 and S_1 are able to jointly obtain the comparison result $[\![b_i]\!]$ by evaluating DICF keys with Δx_i, where $b_i := 1$ if $t_i - x_{k_i}$ is positive and $b_i := 0$ otherwise.

Path evaluation. S_0 and S_1 first generate random, unique and non-zero masks r_i together for each decision node $P_i \in \mathcal{P}$, and then locally compute the left out-going edge cost $[\![e_{i,0}]\!] := [\![(1 - b_i) \cdot r_i]\!]$ and

Constant-round Evaluation Protocol $\Pi_{\mathsf{eval}}^{\mathsf{const}}$

Initialization:

- S_0 and S_1 agree on a random seed $\eta_0 \leftarrow \{0,1\}^\lambda$;
- S_1 and S_2 agree on a random seed $\eta_1 \leftarrow \{0,1\}^\lambda$;
- S_2 and S_0 agree on a random seed $\eta_2 \leftarrow \{0,1\}^\lambda$.

Offline phase:

- Upon initialization, the player S_2 does:
  - **For** $\tau := 2^{\ell-1} - 1$, $i := 0$ to $\widetilde{m}_c - 1$:
    - Generate $\rho_i \leftarrow \mathbb{Z}_{2^\ell}$;
    - Set $\mathcal{K}_{\rho_i}^{(0)}, \mathcal{K}_{\rho_i}^{(1)} \leftarrow \mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\rho_i,0}^{\mathsf{IC}})$ for the function $f_{0,\tau,\rho_i,0}^{\mathsf{IC}} : \mathbb{Z}_{2^\ell} \to \mathbb{Z}_{2^\lambda}$;
  - Send $(\mathsf{sid}, \mathcal{K}_{\rho_i}^{(0)})_{i \in \mathbb{Z}_{\widetilde{m}_c}}$ to S_0, $(\mathsf{sid}, \mathcal{K}_{\rho_i}^{(1)})_{i \in \mathbb{Z}_{\widetilde{m}_c}}$ to S_1;

Online phase:

- Upon receiving $(\mathsf{Eval}, \mathsf{sid})$ from the environment $\mathcal{Z}$, the player $S_j, j \in \mathbb{Z}_3$ does:
  - **For** $i := 0$ to $\widetilde{m}_c - 1$:
    - Set $w_{i,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i)$, $w_{i,j+2} \leftarrow \mathsf{PRF}_{\eta_{j+2}}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i)$;
    - Set $w'_{i,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_n}(\mathsf{sid}, i)$, $w'_{i,j+2} \leftarrow \mathsf{PRF}_{\eta_{j+2}}^{\mathbb{Z}_n}(\mathsf{sid}, i)$;
      // feature selection
    - If $j = 2$, set $k_i^{(j)} := 0$ and $t_i^{(j)} := \rho_i$;
    - Set $k_i^{(j)} := k_i^{(j)} + w'_{i,j} - w'_{i,j+2} \pmod n$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{x}^{(j+1)}, k_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
      // comparison
    - Set $\Delta x_i^{(j)} := t_i^{(j)} - x_{k_i}^{(j)} + w_{i,j} - w_{i,j+2} \pmod{2^\ell}$;
  - Send $(\mathsf{sid}, \Delta\mathbf{x}^{(j)})$ to S_0 and S_1;
- Upon receiving $(\mathsf{sid}, \Delta\mathbf{x}^{(1-j)})$ from S_{1-j}, $(\mathsf{sid}, \Delta\mathbf{x}^{(2)})$ from S_2, the player $S_j, j \in \{0,1\}$ does:
  - **For** $i := 0$ to $\widetilde{m}_c - 1$, set:
    - $\Delta x_i := \sum_{k=0}^{2} \Delta x_i^{(k)} \pmod{2^\ell}$;
    - $b_i^{(j)} \leftarrow \mathsf{DICF.Eval}_{0,\tau}^{\mathsf{IC}}(j, \mathcal{K}_{\rho_i}^{(j)}, \Delta x_i)$;
      // path evaluation
    - $r_i \leftarrow \mathsf{PRF}_{\eta_0}^{\mathbb{Z}_{2^\lambda}}(\mathsf{sid}, i)$;
    - $e_{i,0}^{(j)} := (1 - j - b_i^{(j)}) \cdot r_i$ and $e_{i,1}^{(j)} := b_i^{(j)} \cdot r_i$;
  - **For** $\delta \leftarrow \mathsf{PRF}_{\eta_0}^{\mathbb{Z}_{\widetilde{m}_c+1}}(\mathsf{sid}, 0)$, $i := 0$ to $\widetilde{m}_c$:
    - Sum up the share of edge costs along i-th leaf node's path to get $c_i^{(j)}$, set $\hat{c}_i^{(j)} := c_{i-\delta \pmod{\widetilde{m}_c+1}}^{(j)}$;
    - Set $w_i^{(0)} \leftarrow \mathsf{PRF}_{\eta_0}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i, 0)$, $w_i^{(1)} \leftarrow \mathsf{PRF}_{\eta_0}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i, 1)$;
    - Set $\hat{v}^{(j)} := v_{i-\delta \pmod{\widetilde{m}_c+1}}^{(j)} - w_i^{(j)} \pmod{2^\ell}$;
  - Set $\hat{\mathbf{c}}^{(j)} := (\hat{c}_i^{(j)})_{i \in \mathbb{Z}_{\widetilde{m}_c+1}}$, $\hat{\mathbf{v}}^{(j)} := (\hat{v}_i^{(j)})_{i \in \mathbb{Z}_{\widetilde{m}_c+1}}$;
  - Send $(\mathsf{sid}, \hat{\mathbf{c}}^{(j)}, \hat{\mathbf{v}}^{(j)})$ to S_2;
- Upon receiving $(\mathsf{sid}, \hat{\mathbf{c}}^{(0)}, \hat{\mathbf{v}}^{(0)})$ from S_0, $(\mathsf{sid}, \hat{\mathbf{c}}^{(1)}, \hat{\mathbf{v}}^{(1)})$ from S_1, the player S_2 does:
  - **For** $p := 0$ to $\widetilde{m}_c$, if $\hat{c}_p^{(0)} + \hat{c}_p^{(1)} = 0 \pmod{2^\lambda}$:
    - Set $(\mathcal{K}_p^{(0)}, \mathcal{K}_p^{(1)}) \leftarrow \mathsf{DPF.Gen}(1^\lambda, f_{p,1})$ for the point function $f_{p,1} : \mathbb{Z}_{\widetilde{m}_c+1} \to \mathbb{Z}_{2^\ell}$;
    - Send $(\mathsf{sid}, \mathcal{K}_p^{(0)})$ to S_0, $(\mathsf{sid}, \mathcal{K}_p^{(1)})$ to S_1;
    - Return $y^{(3)} := \hat{v}_p^{(0)} + \hat{v}_p^{(1)} \pmod{2^\ell}$ to the receiver R.
- Upon receiving $(\mathsf{sid}, \mathcal{K}_p^{(j)})$ from S_2, $S_j, j \in \{0,1\}$ does:
  - Set $(\beta_i^{(j)})_{i \in \mathbb{Z}_{\widetilde{m}_c+1}} \leftarrow \mathsf{DPF.EvalAll}(j, \mathcal{K}_p^{(j)})$;
  - Set $y^{(j)} := \sum_{i=0}^{\widetilde{m}_c} (w_i^{(0)} + w_i^{(1)}) \cdot \beta_i^{(j)} \pmod{2^\ell}$;
  - Return $y^{(j)}$ to the receiver R.

Figure 10: Constant-round Evaluation Protocol $\Pi_{\mathsf{eval}}^{\mathsf{const}}$ in the $\mathcal{F}_{\mathsf{sot}}$-hybrid model. For simplicity, we omit $\pmod 3$ from the expressions with $j \in \mathbb{Z}_3$.

right out-going edge cost $[\![e_{i,1}]\!] := [\![b_i \cdot r_i]\!]$ of the node P_i. Subsequently, as shown in Fig. 8, for each leaf node $v_i \in \mathbf{v}$, S_0 and S_1 jointly compute its corresponding path cost $[\![c_i]\!]$ by summing up the edge costs along the path from root to $v_i \in \mathbf{v}$. All path costs form a shared vector $[\![\mathbf{c}]\!] := ([\![c_i]\!])_{i \in \mathbb{Z}_{\widetilde{m}_c+1}}$, whose only one entry with 0 value indicates the position of the classification result. To obliviously select the classification result according to $[\![\mathbf{c}]\!]$, S_0 and S_1 generate a random offset $\delta \leftarrow \mathbb{Z}_{\widetilde{m}_c+1}$ together,

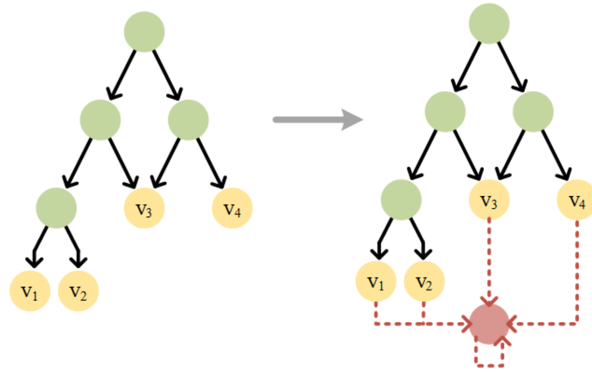


Figure 11: Sparse DAG depth-padding.

and jointly cyclic-shift $\mathbf{c}$ and $\mathbf{v}$ to the left δ positions to get $\hat{\mathbf{c}}$ and $\mathbf{v}'$; then generate a random vector $\mathbf{w} := (w_i)_{i\in\widetilde{m}_c+1} \leftarrow (\mathbb{Z}_{2^\ell})^{\widetilde{m}_c+1}$ together, and jointly compute $\hat{\mathbf{v}} := \mathbf{v}' - \mathbf{w}$. After that, they open $\hat{\mathbf{c}}$ and $\hat{\mathbf{v}}$ to S_2. Upon reconstructing $\hat{\mathbf{c}}$ and $\hat{\mathbf{v}}$, S_2 generates a pair of DPF keys for the point function $f_{p,1}(x)$ according to the position p of $\hat{c}_p = 0$, and distributes DPF keys to S_0 and S_1. Finally, S_0, S_1 evaluate DPF keys on the random vector $\mathbf{w}$ as

$$[\![w_p]\!] := \sum_{i=0}^{\widetilde{m}_c} w_i \cdot [\![f_{p,1}(i)]\!] \pmod{2^\ell}$$

and then return $[\![w_p]\!]$ to the receiver, while S_2 directly returns $\hat{v}_p$. It is easy to see, the result $v_p = w_p + \hat{v}_p \pmod{2^\ell}$.

Correctness. The correctness of our evaluation would not hold only when more than one entry of the path-cost vector $\mathbf{c}$ is equal to 0, e.g., $c_7 := r_0 + r_2 + r_6 = 0$ in Fig. 8. Because the party S_2 who opens the shifted $\hat{\mathbf{c}}$ cannot select the correct result position from multiple 0 entries. Except for the result path, we find that the last non-zero edge cost of each path is unique. Thus we can say that this unique edge cost "determines" the corresponding path cost. For example, r_6 "determines" the value of $c_7 := r_0 + r_2 + r_6$ in Fig. 8, and $c_7 = 0$ only if $r_6 = -r_0 - r_2$. Therefore, the probability of appearing any extra 0 entry in the path-cost vector $\mathbf{c}$, i.e., the failure probability of our constant-round protocol, is $(2^{d-1} - 1)/2^\lambda$. It is negligible in λ.

Security. We show the security of our constant-round protocol ($\Pi_{\mathsf{os}}^{\mathsf{const}}$, $\Pi_{\mathsf{eval}}^{\mathsf{const}}$) with the following theorem, and its proof can be found in section C of the supplemental material.

Theorem 3. *Let $\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}$ be a secure function secret sharing scheme for $f^{\mathsf{IC}}_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}(x) : \mathbb{Z}_{2^\ell} \mapsto \mathbb{Z}_{2^\lambda}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}$ be a secure function secret sharing scheme for point function $f_{\alpha,\beta}(x) : \mathbb{Z}_{\widetilde{m}_c+1} \mapsto \mathbb{Z}_{2^\ell}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\ell}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_n} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_n$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\lambda}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{\widetilde{m}_c+1}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A})$. The protocols $\Pi_{\mathsf{os}}^{\mathsf{const}}$ as described in Fig. 9 and $\Pi_{\mathsf{eval}}^{\mathsf{const}}$ as described in Fig. 10 UC-realize $\mathcal{F}^3_{\mathsf{bp}}$ as described in Fig. 2 in the $\mathcal{F}_{\mathsf{sot}}$-hybrid model against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage at most*

$$\begin{aligned}
&3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + 3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A}) \\
&+ \widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) + \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \\
&+ (\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) + \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A}) \\
&+ 2(\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \ .
\end{aligned}$$

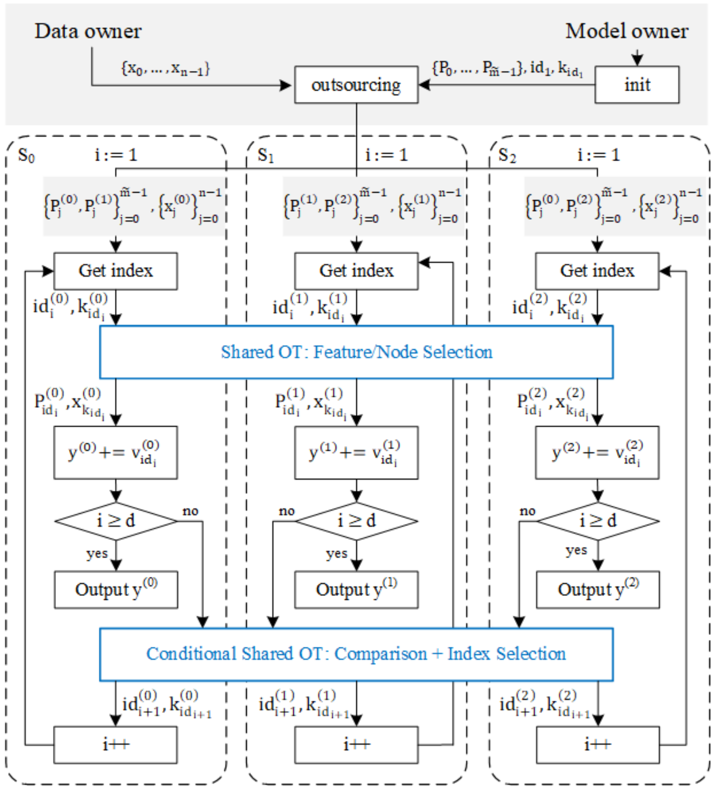


Figure 12: Overview of our linear-round protocol.

6.2 Linear-Round Protocol

Our linear-round protocol supports sparse tree and BP evaluation. To hide the model structure, we introduce only one dummy node instead of transforming the sparse decision tree into a full tree, i.e. $\widetilde{m} = m + 1$. Let the dummy node point to itself and all leaf nodes point to it as shown in Fig. 11. The main idea is that, during privacy-preserving evaluation, once a sink node is reached, servers will obliviously access this dummy node (repeatedly) until the protocol reaches d steps. Thus, the length of evaluation path is always d.

We use a vector to describe this padded model, which includes all kinds of nodes. Without confusion, we also denote it as $\mathcal{P}$. Each $P_i \in \mathcal{P}$ consists of the index $\mathcal{I}_i^{\mathsf{left}}$ and the input selection index $\mathcal{J}_i^{\mathsf{left}}$ of its left child, the index $\mathcal{I}_i^{\mathsf{right}}$ and the input selection index $\mathcal{J}_i^{\mathsf{right}}$ of its right child, a threshold value t_i and a classification value v_i of P_i. If P_i represents the dummy node, $\mathcal{I}_i^{\mathsf{left}}$ and $\mathcal{I}_i^{\mathsf{right}}$ take the value of the index of dummy node $\widetilde{m}$, $\mathcal{J}_i^{\mathsf{left}}$ and $\mathcal{J}_i^{\mathsf{right}}$ take random values, and v_i is equal to 0. If P_i represents a decision node, v_i is dummy data such that $v_i = 0$. If P_i represents a sink node, $\mathcal{I}_i^{\mathsf{left}}$, $\mathcal{I}_i^{\mathsf{right}}$, $\mathcal{J}_i^{\mathsf{left}}$ and $\mathcal{J}_i^{\mathsf{right}}$ are the same dummy data as the dummy node. Since there only is one leaf node in a path, and only if v belongs to a leaf node the value of v is non-zero, the accumulation of v of all nodes in the evaluation path is exactly equal to the classification value of the reached leaf node.

Our linear-round protocol requires $2d$ rounds. Referring to the example in Fig. 12, for i-th step in the evaluation, servers first obliviously fetch the "current node" P_{id_i} and the appropriate feature x_{k_i} in the Round 1. Then compute:

$$y := y + v_i,$$
$$c \leftarrow (x_{k_i} < t_i).$$

and indicates the next node index is $\mathcal{I}_i^{\mathsf{left}}$ $(c = 1)$ or $\mathcal{I}_i^{\mathsf{right}}$ $(c = 0)$ in the Round 2. After repeating the above process d times, $\langle y \rangle$ is open to receiver as the evaluation result.

Outsourcing. For linear-round protocol, the data owner outsourcing protocol is identical to our constant-round scheme, but the model owner outsourcing protocol is different. As described in Fig. 13, Outsourcing Protocol $\Pi^{\mathsf{linear}}_{\mathsf{os}}$

- Upon receiving (Model, $\mathsf{sid}, \mathcal{P}$) from the environment $\mathcal{Z}$, the model owner M:
  - Build the position mapping, denote i-th element as $P_i := \{\mathcal{I}_i^{\mathsf{left}}, \mathcal{I}_i^{\mathsf{right}}, \mathcal{J}_i^{\mathsf{left}}, \mathcal{J}_i^{\mathsf{right}}, t_i, v_i\}$;
  - **For** $i := 0$ to $\widetilde{m} - 1$, set:
    - $\mathcal{I}_i^{\mathsf{left},(0)}, \mathcal{I}_i^{\mathsf{left},(1)}, \mathcal{I}_i^{\mathsf{right},(0)}, \mathcal{I}_i^{\mathsf{right},(1)} \leftarrow \mathbb{Z}_{\widetilde{m}}$;
    - $\mathcal{I}_i^{\mathsf{left},(2)} := \mathcal{I}_i^{\mathsf{left}} - \mathcal{I}_i^{\mathsf{left},(0)} - \mathcal{I}_i^{\mathsf{left},(1)} \pmod{\widetilde{m}}$;
    - $\mathcal{I}_i^{\mathsf{right},(2)} := \mathcal{I}_i^{\mathsf{right}} - \mathcal{I}_i^{\mathsf{right},(0)} - \mathcal{I}_i^{\mathsf{right},(1)} \pmod{\widetilde{m}}$;
    - $\mathcal{J}_i^{\mathsf{left},(0)}, \mathcal{J}_i^{\mathsf{left},(1)}, \mathcal{J}_i^{\mathsf{right},(0)}, \mathcal{J}_i^{\mathsf{right},(1)} \leftarrow \mathbb{Z}_n$;
    - $\mathcal{J}_i^{\mathsf{left},(2)} := \mathcal{J}_i^{\mathsf{left}} - \mathcal{J}_i^{\mathsf{left},(0)} - \mathcal{J}_i^{\mathsf{left},(1)} \pmod{n}$;
    - $\mathcal{J}_i^{\mathsf{right},(2)} := \mathcal{J}_i^{\mathsf{right}} - \mathcal{J}_i^{\mathsf{right},(0)} - \mathcal{J}_i^{\mathsf{right},(1)} \pmod{n}$;
    - $t_i^{(0)}, t_i^{(1)}, v_i^{(0)}, v_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - $t_i^{(2)} := t_i - t_i^{(0)} - t_i^{(1)} \pmod{2^\ell}$;
    - $v_i^{(2)} := v_i - v_i^{(0)} - v_i^{(1)} \pmod{2^\ell}$;
  - Set $\mathsf{id}_1 := 0$ and k_1 to the feature index of the root;
  - Set $\mathsf{id}_1^{(0)}, \mathsf{id}_1^{(1)} \leftarrow \mathbb{Z}_{\widetilde{m}}$, $k_1^{(0)}, k_1^{(1)} \leftarrow \mathbb{Z}_n$;
  - Set $\mathsf{id}_1^{(2)} := \mathsf{id}_1 - \mathsf{id}_1^{(0)} - \mathsf{id}_1^{(1)} \pmod{\widetilde{m}}$;
  - Set $k_1^{(2)} := k_1 - k_1^{(0)} - k_1^{(1)} \pmod{n}$;
  - **For** $j \in \mathbb{Z}_3$, send $(\mathcal{P}^{(j)}, \mathcal{P}^{(j+1 \pmod 3)}, \mathsf{id}_1^{(j)}, k_1^{(j)})$ to S_j;
- Upon receiving (Data, $\mathsf{sid}, \mathbf{x}$) from the environment $\mathcal{Z}$, the data owner D:
  - **Foreach** feature $x_i \in \mathbf{x}$:
    - Generate $x_i^{(0)}, x_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - Set $x_i^{(2)} := x_i - x_i^{(0)} - x_i^{(1)} \pmod{2^\ell}$;
  - **For** $j \in \mathbb{Z}_3$, send $(\mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{x}^{(j+1 \pmod 3)})$ to S_j.

Figure 13: Outsourcing Protocol $\Pi^{\mathsf{linear}}_{\mathsf{os}}$.

the model owner M generates replicated secret sharing of $\mathcal{P}$ among three servers. In order to make servers aware of the evaluation entry, M shares the element index id_1 and the feature selection index k_{id_1} of the root node to three servers in $(3,3)$-additive secret sharing.

Evaluation. For i-th step in the evaluation, with the secret shared element index $\langle k_i \rangle$ and feature index $\langle \mathsf{id}_i \rangle$, three servers invoke our shared OT protocol to fetch the feature $\langle x_{k_i} \rangle$ and the element $\langle P_{\mathsf{id}_i} \rangle$ in parallel. For readability, we describe our protocol $\Pi^{\mathsf{linear}}_{\mathsf{eval}}$ in the $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$-hybrid model in Fig. 14. Then servers jointly sum the v_{id_i} for path evaluation, which is a free operation in our protocol. If $i < d$, servers invoke the conditional shared OT protocol to compare the threshold and corresponding feature of current node and obtain the element index $\langle k_{i+1} \rangle$ and feature index $\langle \mathsf{id}_{i+1} \rangle$ of next node; then repeat the above operation. If $i = d$, each server returns its share of y to the requester, who is able to reconstruct the classification result locally.

Security. We show the security of our linear-round protocol ($\Pi^{\mathsf{linear}}_{\mathsf{os}}$, $\Pi^{\mathsf{linear}}_{\mathsf{eval}}$) with the following theorem, and its proof can be found in section D of the supplemental material.

Theorem 4. *The protocol $\Pi^{\mathsf{linear}}_{\mathsf{os}}$ as described in Fig. 13 and $\Pi^{\mathsf{linear}}_{\mathsf{eval}}$ as described in Fig. 14 UC-realizes $\mathcal{F}^3_{\mathsf{bp}}$ as described in Fig. 2 in the $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$-hybrid model against semi-honest adversaries who can statically corrupted up to 1 server.*

7 Implementation and Benchmarks

The proposed constant-round scheme and linear-round scheme are implemented in C++. The DCF and DPF schemes are improved from [23]. Since Ma *et al.* [11] did not release their source code, we re-implement their scheme using AES-NI and EMP-toolkits [24]. In addition, the state-of-the-art constant-round protocols are adopted from the open source of [7] for performance comparison. Our benchmarks are executed on a desktop with Intel(R) Core i7 8700 CPU @ 3.2 GHz running Ubuntu 18.04.2 LTS; with 6 CPUs, 32 GB Memory and 1TB SSD. Their network environments are simulated: local-area network (LAN, RTT: 0.1ms, bandwidth: 1Gbps), metropolitan-area network (MAN, RTT: 6ms, bandwidth: 100Mbps), and wide-area network (WAN, RTT: 80ms, bandwidth: 40Mbps).

Linear-round Evaluation Protocol $\Pi_{\mathsf{eval}}^{\mathsf{linear}}$

- Upon receiving $(\mathsf{Eval}, \mathsf{sid})$ from $\mathcal{Z}$, the player $S_j, j \in \mathbb{Z}_3$ does:
  - Set $y^{(j)} := 0$;
  - **For** $i := 1$ to d:
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{x}^{(j+1)}, k_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$, get $x_{k_i}^{(j)}$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathcal{P}^{(j)}, \mathcal{P}^{(j+1)}, \mathsf{id}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{\widetilde{m},*}$, get $P_{id_i}^{(j)} := (\mathcal{I}_{id_i}^{\mathsf{left},(j)}, \mathcal{I}_{id_i}^{\mathsf{right},(j)}, \mathcal{J}_{id_i}^{\mathsf{left},(j)}, \mathcal{J}_{id_i}^{\mathsf{right},(j)}, t_{id_i}^{(j)}, v_{id_i}^{(j)})$[a]
    - Set $y^{(j)} := y^{(j)} + v_{id_i}^{(j)} \pmod{2^\ell}$;
    - **If** $i \geq d$, return $y^{(j)}$ to the receiver R and **break**;
    - Obliviously fetch $\mathsf{id}_{i+1}^{(j)}$ and $k_{i+1}^{(j)}$ by sending:[b]
      - $(\textsc{Sel}, \mathsf{sid}, (\mathcal{I}_{id_i}^{\mathsf{left},(j)}, \mathcal{I}_{id_i}^{\mathsf{right},(j)}), (x_{k_i}^{(j)}, t_{id_i}^{(j)}))$ to $\mathcal{F}_{\mathsf{csot}}^{\log \widetilde{m},\ell}$;
      - $(\textsc{Sel}, \mathsf{sid}, (\mathcal{J}_{id_i}^{\mathsf{left},(j)}, \mathcal{J}_{id_i}^{\mathsf{right},(j)}), (x_{k_i}^{(j)}, t_{id_i}^{(j)}))$ to $\mathcal{F}_{\mathsf{csot}}^{\log n,\ell}$.

[a] Here, we invoke $\mathcal{F}_{\mathsf{sot}}^{\widetilde{m},*}$ for readability. In practice, the servers select each item for the element $P_{id_i}^{(j)}$ by the same DPF keys but different output bit-lengths of evaluation. More specifically, the DPF key generator will generate several correction words of different bit-lengths for the last conversion operation as shown in [19]; and the DPF evaluators will perform the conversion operation multiple times to get several outputs of different bit-lengths using these correction words. And all selections of i-th step are performed in the same round, including the feature selection of the previous row.

[b] Servers select id_{i+1} and k_{i+1} based on the same DCF keys and during the same round.

Figure 14: Linear-round Evaluation Protocol $\Pi_{\mathsf{eval}}^{\mathsf{linear}}$ in the $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$-hybrid model.

Our experiment uses datasets from the UCI machine learning repository [25], which consists of Iris, Wine (chemical analysis), Linnerud (physical exercise performance), Breast (cancer), Digits, Spambase, Diabetes, and Boston (housing value). Their concrete parameters are shown in Table 2. We set secure parameter λ to 128, feature bit-length ℓ to 64. Note that the performance results of the related works, e.g., MTZC [11], are slightly different from that presented in the original papers due to different implementation and experiment environment. The main overhead of the offline phase of our protocol is to generate the FSS key. Compared with MTZC [11], as shown in Table. 3, our protocol is slightly slower for small DAG models, while it is about 4X faster for big DAG models.

Fig.15 illustrates the online runtime comparison between our two protocols and the related works. The results are taken as the average of 10 evaluations. We fail to obtain the evaluation results for Diabetes and Boston models for our constant-round protocol and MTZC outsourcing protocols, as both protocols require complete-tree padding. For depth $d = 28, 30$ trees, complete decision tree padding would cause the memory out of computer capacity.

In a network environment with higher bandwidth and lower latency such as the LAN setting, our linear-round protocol runs much more faster than the state-of-the-arts. More precisely, our linear-round protocol is up to 15X faster than the others in the LAN setting. In a network environment with lower bandwidth and higher latency such as the WAN setting, our constant-round protocol outperforms the state-of-the-art protocols. In particular, our constant-round protocol is up to 10X faster than the others in the WAN setting.

Our constant-round protocol has low round complexity (3-round), high communication (but better

Table 2: Parameters of the models in the UCI dataset.

Decision Tree	Features	Depth	Nodes
Iris	4	4	7
Wine	7	5	11
Linnerud	3	6	19
Breast	12	7	21
Digits	47	15	168
Spambase	57	17	58
Diabetes	10	28	393
Boston	13	30	425

Table 3: Offline phase running time comparison (ms) between our (2d-1)-round protocol and MTZC [11] in the outsourced setting. (Network setting: MAN (100Mbps/6ms RTT) and WAN (40Mbps/80ms RTT)

		Linnerud	Breast	Digits	Spambase
MAN	MTZC	10.319	11.190	158.6	177.5
	Ours(Linear)	17.433	23.103	43.423	45.43
WAN	MTZC	87.541	91.56	615.9	879.9
	Ours(Linear)	108.45	109.382	145.55	209.376

than all other known constant-round protocols), and high computation; whereas, our linear-round protocol has linear round complexity (2d-1 rounds), low communication, and low computation. Therefore, our constant-round protocol outperforms all the other protocols in the WAN setting. Our linear-round protocol outperforms all the other protocols in the LAN setting. In terms of the MAN setting, our linear-round protocol has an increasing advantage along with the increase of the tree size due to its low communication.

8 Related Work

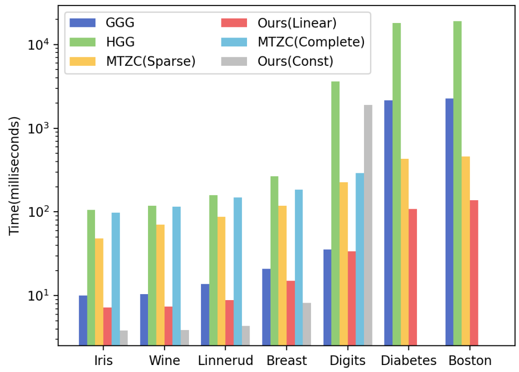
There has been a huge literature in private BP and/or decision tree evaluation. The first work is proposed by Ishai and Paskin [1]. They evaluate a BP on encrypted input via homomorphic public-key cryptosystem, and require $O(md)$ communication. It is impractical for cases with a large number of input features, like medical diagnosis. And their protocol does not include comparison in each non-sink node.

Later, many evaluation protocols are proposed also with constant communication round. Brikell *et al.* [2] present a private diagnosis system based on BP model. They implement privately feature selection with *additive HE* (AHE) and *oblivious transfer* (OT), and transform the whole BP into a secure program consisiting of GCs representing permuted nodes to evaluate comparisons. [4] treats a decision tree as a high-degree polynomial with a priori fixed multiplicative depth and evaluate the polynomial through costly *full HE* (FHE) to obtain result. [5] gets rid of FHE by using DGK protocol based on AHE for comparison and OT for leaf node selection. But [5] requires a complete tree (with dummy nodes) and permuting it. [6] improves [5] by a new "path cost" approach, which is a linear function for each path and determines whether a leaf node contains the classification result. Their protocol is purely based on AHE, without introducing dummy nodes. Obviously, [5] and [6] take advantage of the properties of the tree structure, thus no longer support BP evaluation. [7] systematically reviews prior constant-round solutions and proposes a modular construction from three constant-round sub-protocols: (a) private feature selection, (b) secure comparison, and (c) oblivious path evaluation. [7] also identifies novel combinations of these linear sub-protocols that provide better tradeoffs.

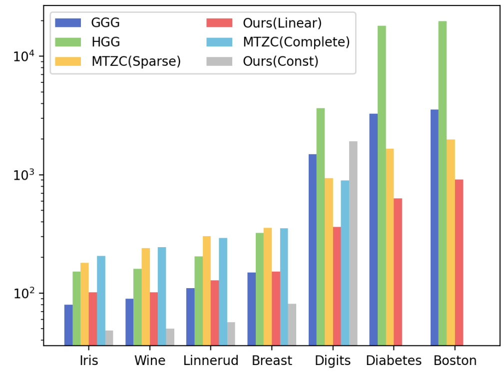
On the other hand, constant-round protocols above always require the client to have at least linear computation in the model size m, which is not friendly to weak client with limited computational resource. Thus, researchers are attracted to pursue new solutions with sublinear computation complexity for client, i.e., the parties can only adaptively perform necessary feature selections and comparisons along with the evaluation path. The main idea is to obliviously select only one decision node for comparison at each layer of the DAG via either OT or ORAM, such as [8] and [9]. The dependence of the current selection on previous comparison results leads to the round complexity of protocol is usually linear in the length d of the longest path.

Recently, the outsourcing extension is considered in private BP and/or decision tree evaluation[1]. The protocol of [10] is based on boolean secret sharing. It requires (padded) full decision trees, and includes m secure matrix multiplications for input selection, $2^{d-1}-1$ bit-wise comparison with SS and $O(2^d)$ multiplications for path evaluation, which needs $O(d)$ rounds and $O(mn)$ communication. [14] is inspired by [10], and use additive secret sharing. [14] introduces a standard modulus conversion after bit-wise comparison and follows the path cost computation of [6]. The protocol of [14] has the

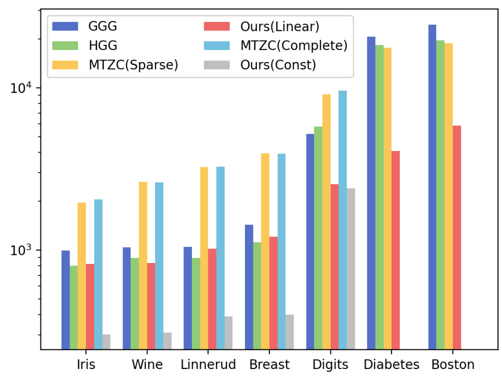
[1]We consider the secure outsourcing without the leakage of the index mapping between decision nodes and input features. [15] and [13] do not meet this condition.



(a) LAN(1Gbps/0.1ms)



(b) MAN(100Mbps/6ms)



(c) WAN(40Mbps/80ms)

Figure 15: Online runtime (in different log scales) in LAN/MAN/WAN (bandwidth/RTT) setting.Ours(Linear)/Ours(Const) refers to our linear-round/constant-round protocol. MTZC(sparse) refers to the sparse tree variant of [11]; MTZC(complete) refers to their outsourcing variant.

same communication complexity as [10]. The state-of-the-art work of outsourced evaluation protocol is from [11]. It presents a key management and uses conditional OT to reduce the communication cost, and reaches $2d-1$ rounds and $O(d)$ communication in online phase. The outsourced protocol of [11] requires both parties refresh their shared decision tree for each evaluation, and only support complete decision tree. They lead to $O(2^d)$ offline communication. In addition, none of the above outsourced evaluation protocol support privacy-preserving BP evaluation.

9 Concluding Remarks

We presented a 3-server MPC platform for outsourced private decision tree and BP evaluation. For uniformity, we assume each BP decision node also has a comparison; however, it can be easily removed to adapt to any other binary decision diagram. Our key building block is a lightweight 1-out-of-N shared OT protocol with logarithmic communication. Unlike [17], we utilize the DPF scheme in a novel way such that the ORAM functionality is achieved without the need of oblivious PRF evaluation via MPC. Our linear-round outsourced private decision tree evaluation protocol achieves logarithmic communication in both online and offline; yet, it is unknown if there exists a constant-round protocol with logarithmic overall communication. We leave this as an open problem.

References

[1] Y. Ishai and A. Paskin, "Evaluating branching programs on encrypted data," in *TCC 2007*, vol. 4392 of *LNCS*, (Amsterdam, The Netherlands), pp. 575–594, Springer, Feb. 21–24, 2007.

[2] J. Brickell, D. E. Porter, V. Shmatikov, and E. Witchel, "Privacy-preserving remote diagnostics," in *ACM CCS 2007*, (Alexandria, Virginia, USA), pp. 498–507, ACM Press, Oct. 28–31, 2007.

[3] M. Barni, P. Failla, V. Kolesnikov, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Secure evaluation of private linear branching programs with medical applications," in *ESORICS 2009*, vol. 5789 of *LNCS*, (Saint-Malo, France), pp. 424–439, Springer, Sept. 21–23, 2009.

[4] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *NDSS 2015*, (San Diego, CA, USA), The Internet Society, Feb. 8–11, 2015.

[5] D. J. Wu, T. Feng, M. Naehrig, and K. E. Lauter, "Privately evaluating decision trees and random forests," *PoPETs*, vol. 2016, pp. 335–355, Oct. 2016.

[6] R. K. H. Tai, J. P. K. Ma, Y. Zhao, and S. S. M. Chow, "Privacy-preserving decision trees evaluation via linear functions," in *ESORICS 2017, Part II*, vol. 10493 of *LNCS*, (Oslo, Norway), pp. 494–512, Springer, Sept. 11–15, 2017.

[7] Á. Kiss, M. Naderpour, J. Liu, N. Asokan, and T. Schneider, "SoK: Modular and efficient private decision tree evaluation," *PoPETs*, vol. 2019, pp. 187–208, Apr. 2019.

[8] M. Joye and F. Salehi, "Private yet efficient decision tree evaluation," in *DBSec 2018*, vol. 10980, (Bergamo, Italy), pp. 243–259, Springer, July 16–18 2018.

[9] A. Tueno, F. Kerschbaum, and S. Katzenbeisser, "Private evaluation of decision trees using sublinear cost," *PoPETs*, vol. 2019, pp. 266–286, Jan. 2019.

[10] M. De Cock, R. Dowsley, C. Horst, R. Katti, A. C. A. Nascimento, W. Poon, and S. Truex, "Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation," *IEEE Trans. Dependable Secur. Comput.*, vol. 16, pp. 217–230, Feb. 2017.

[11] J. P. K. Ma, R. K. H. Tai, Y. Zhao, and S. S. Chow, "Let's stride blindfolded in a forest: Sublinear multi-client decision trees evaluation," in *NDSS 2021*, (San Diego, CA, USA), The Internet Society, Feb. 21-25, 2021.

[12] X. Wang, T.-H. H. Chan, and E. Shi, "Circuit ORAM: On tightness of the Goldreich-Ostrovsky lower bound," in *ACM CCS 2015*, (Denver, CO, USA), pp. 850–861, ACM Press, Oct. 12–16, 2015.

[13] A. Aloufi, P. Hu, H. W. H. Wong, and S. S. M. Chow, "Blindfolded evaluation of random forests with multi-key homomorphic encryption," *IEEE Trans. Dependable Secur. Comput.*, vol. 18, no. 4, pp. 1821–1835, 2019.

[14] Y. Zheng, H. Duan, and C. Wang, "Towards secure and efficient outsourcing of machine learning classification," in *ESORICS 2019, Part I*, vol. 11735 of *LNCS*, (Luxembourg), pp. 22–40, Springer, Sept. 23–27, 2019.

[15] L. Liu, J. Su, R. Chen, J. Chen, G. Sun, and J. Li, "Secure and fast decision tree evaluation on outsourced cloud data," in *ML4CS 2019*, (Xi'an, China), pp. 361–377, Springer, Sept. 19–21, 2019.

[16] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing," in *EUROCRYPT 2015, Part II*, vol. 9057 of *LNCS*, (Sofia, Bulgaria), pp. 337–367, Springer, Apr. 26–30, 2015.

[17] J. Doerner and a. shelat, "Scaling ORAM for secure computation," in *ACM CCS 2017*, (Dallas, TX, USA), pp. 523–535, ACM Press, Oct. 31 – Nov. 2, 2017.

[18] P. Bunn, J. Katz, E. Kushilevitz, and R. Ostrovsky, "Efficient 3-party distributed ORAM," in *SCN 20*, vol. 12238 of *LNCS*, (Amalfi, Italy), pp. 215–232, Springer, Sept. 14–16, 2020.

[19] E. Boyle, N. Chandran, N. Gilboa, D. Gupta, Y. Ishai, N. Kumar, and M. Rathee, "Function secret sharing for mixed-mode and fixed-point secure computation," in *EUROCRYPT 2021, Part II*, vol. 12697 of *LNCS*, (Zagreb, Croatia), pp. 871–900, Springer, Oct. 17–21, 2021.

[20] E. Boyle, N. Gilboa, and Y. Ishai, "Function secret sharing: Improvements and extensions," in *ACM CCS 2016*, (Vienna, Austria), pp. 1292–1303, ACM Press, Oct. 24–28, 2016.

[21] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *FOCS'01*, pp. 136–145, IEEE, 2001.

[22] R. Cramer, I. Damgård, and Y. Ishai, "Share conversion, pseudorandom secret-sharing and applications to secure computation," in *TCC 2005* (J. Kilian, ed.), vol. 3378 of *LNCS*, (Cambridge, MA, USA), pp. 342–362, Springer, Feb. 10–12, 2005.

[23] F. Wang, "Function Secret Sharing (FSS) Library." `https://github.com/frankw2/libfss`, 2019.

[24] X. Wang, A. J. Malozemoff, and J. Katz, "EMP-toolkit: Efficient MultiParty computation toolkit." `https://github.com/emp-toolkit`, 2016.

[25] D. Dua and C. Graff, "UCI machine learning repository." `http://archive.ics.uci.edu/ml`, 2017.

[26] V. Kolesnikov, N. Matania, B. Pinkas, M. Rosulek, and N. Trieu, "Practical multi-party private set intersection from symmetric-key techniques," in *ACM CCS 2017* (B. M. Thuraisingham, D. Evans, T. Malkin, and D. Xu, eds.), (Dallas, TX, USA), pp. 1257–1272, ACM Press, Oct. 31 – Nov. 2, 2017.

A Proof of Theorem 1

Theorem 1. Let $\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}$ be a secure function secret sharing scheme for point function $f_{\alpha,\beta}(x) : \mathbb{Z}_N \mapsto \mathbb{Z}_{2^\ell}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_N} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_N$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A})$. The protocol $\Pi_{\mathsf{sot}}^{N,\ell}$ as described in Fig. 4 UC-realizes $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$ as described in Fig. 3 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

$$9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A}) + 2 \cdot \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \ .$$

Proof. To prove Thm. 1, we construct a PPT simulator Sim such that no non-uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\mathsf{Exec}_{\Pi_{\mathsf{sot}}^{N,\ell},\mathcal{A},\mathcal{Z}}$ where the parties $\mathcal{S} := \{S_0, S_1, S_2\}$ run protocol $\Pi_{\mathsf{sot}}^{N,\ell}$ in the real world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\mathsf{Exec}_{\mathcal{F}_{\mathsf{sot}}^{N,\ell},\mathsf{Sim},\mathcal{Z}}$ where the parties S_0, S_1, S_2 interact with functionality $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$ in the ideal world, and corrupted parties are controlled by the simulator Sim. Since the protocol is symmetric, we assume S_0 is corrupted for readability.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Since the semi-honest setting, Sim can obtain the correct $i^{(0)}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}$ by simulating a dummy corrupted party S_0 to receive the messages from the environment $\mathcal{Z}$ in the ideal world. Sim simulates the interface of honest parties S_1, S_2. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon initialization, the simulator Sim acts as the honest party S_j, $j \in \{1,2\}$ to do:
  - Generate $\varphi_j \leftarrow \mathbb{Z}_N$;
  - Set $(\mathcal{K}_{\varphi_j}^{(0)}, \mathcal{K}_{\varphi_j}^{(1)}) \leftarrow \mathsf{DPF.Gen}(1^\lambda, f_{\varphi_j,1})$ for the point function $f_{\varphi_j,1} : \mathbb{Z}_N \to \mathbb{Z}_{2^\ell}$;
  - Send $(\mathsf{sid}, \mathcal{K}_{\varphi_j}^{(0)})$ to S_{j+1}, $(\mathsf{sid}, \mathcal{K}_{\varphi_j}^{(1)})$ to S_{j+2};
- The simulator Sim picks random $w_{k,j} \leftarrow \mathbb{Z}_N$ for $k, j \in \mathbb{Z}_3$;
- Upon receiving $(\textsc{Fetch}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1,2\}$ from the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$, the simulator Sim does:

- For $k \in \mathbb{Z}_3$, set $\delta_k^{(j)} := -w_{k,j} + w_{k,j+2} \pmod N$;
- Set $\delta_j^{(j)} := \delta_j^{(j)} - \varphi_j \pmod N$;
- Send $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+1}^{(j)})$ to S_{j+2}, $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+2}^{(j)})$ to S_{j+1} on behave of S_j;

- Upon receiving $(\mathsf{sid}, \delta_0^{(0)}, \delta_1^{(0)})$ from the corrupted S_0 to S_2 and $(\mathsf{sid}, \delta_0^{(0)}, \delta_2^{(0)})$ from the corrupted S_0 to S_1, the simulator Sim does:
  - Send $(\text{FETCH}, \mathsf{sid}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}, i^{(0)})$ to the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$;
  - Compute $\delta_1 := \delta_1^{(0)} + \delta_1^{(1)} + \delta_1^{(2)} \pmod N$;
  - Compute $\delta_2 := \delta_2^{(0)} + \delta_2^{(1)} + \delta_2^{(2)} \pmod N$;
  - Set $(\beta_{k,\varphi_1}^{(1)})_{k \in \mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(1, \mathcal{K}_{\varphi_1}^{(1)})$;
  - Set $(\beta_{k,\varphi_2}^{(0)})_{k \in \mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(0, \mathcal{K}_{\varphi_2}^{(0)})$;
  - Set $y^{(0)} := \sum_{k=0}^{N-1} (x_{k+\delta_1}^{(0)} \cdot \beta_{k,\varphi_1}^{(1)} + x_{k+\delta_2}^{(1)} \cdot \beta_{k,\varphi_2}^{(0)}) \pmod{2^\ell}$;
  - $\zeta_0 \leftarrow \mathsf{PRF}_{\eta_0}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$, $\zeta_2 \leftarrow \mathsf{PRF}_{\eta_2}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$;
  - Compute $y^{(0)} := y^{(0)} + \zeta_0 - \zeta_2 \pmod{2^\ell}$.
  - Send $(\text{RAND}, \mathsf{sid}, y^{(0)})$ to the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$;

Indistinguishability. We assume that the parties S_0, S_1, S_2 communicate with each other via the secure channel functionality $\mathcal{F}_{\mathsf{sc}}$ (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0, \ldots, \mathcal{H}_2$.

Hybrid $\mathcal{H}_0$: It is the real protocol execution $\mathsf{Exec}_{\Pi_{\mathsf{sot}}^{N,\ell}, \mathcal{A}, \mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w_{j,k}\}_{j,k \in \mathbb{Z}_3}$ are picked uniformly random from $\mathbb{Z}_N$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_N}$.

Claim 1. *If $\mathsf{PRF}^{\mathbb{Z}_N} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_N$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_1$ and $\mathcal{H}_0$ are indistinguishable with advantage $\epsilon_1 := 9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A})$.*

Proof. We have changed 3 PRF outputs to uniformly random strings; therefore, the overall advantage is $9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A})$ by hybrid argument via reduction. □

Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_2$:

- For $j \in \{1, 2\}$, the simulator Sim does:
  - Generate $\varphi_j \leftarrow \mathbb{Z}_N$;
  - For $k \in \mathbb{Z}_3$, set $\delta_k^{(j)} := -w_{k,j} + w_{k,j} \pmod N$;
  - Set $\delta_j^{(j)} := \delta_j^{(j)} - \varphi \pmod N$;

instead of

- For $j \in \{1, 2\}$, the honest party S_j does:
  - Generate $\varphi_j \leftarrow \mathbb{Z}_N$;
  - For $k \in \mathbb{Z}_3$, set $\delta_k^{(j)} := i^{(j)} - w_{k,j} + w_{k,j} \pmod N$;
  - Set $\delta_j^{(j)} := \delta_j^{(j)} - \varphi \pmod N$;

Claim 2. *If $\mathsf{DPF}^{\mathbb{Z}_N, \mathbb{Z}_{2^\ell}} := (\mathsf{Gen}, \mathsf{Eval})$ is a secure function secret sharing scheme for point function $f_{\alpha,\beta}(x) : \mathbb{Z}_N \mapsto \mathbb{Z}_{2^\ell}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N, \mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\epsilon_2 := 2 \cdot \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N, \mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$.*

Proof. Note that φ_1, φ_2 are used to generate the DPF keys $\mathcal{K}_{\varphi_1}^{(0)}, \mathcal{K}_{\varphi_1}^{(1)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, f_{\varphi_1,1})$ and $\mathcal{K}_{\varphi_2}^{(0)}, \mathcal{K}_{\varphi_2}^{(1)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, f_{\varphi_2,1})$. The corrupted party S_0 only sees $\mathcal{K}_{\varphi_1}^{(1)}, \mathcal{K}_{\varphi_2}^{(0)}, \{\delta_1^{(j)}, \delta_2^{(j)}\}_{j \in \mathbb{Z}_3}$; therefore, the modification of $\mathcal{K}_{\varphi_1}^{(0)}, \mathcal{K}_{\varphi_2}^{(1)}, \{\delta_0^{(j)}\}_{j \in \mathbb{Z}_3}$ is oblivious to S_0. In the hybrid $\mathcal{H}_1$, we have

- $\delta_1^{(0)} := i^{(0)} - w_{1,0} + w_{1,2}$, $\delta_1^{(1)} := i^{(1)} - w_{1,1} + w_{1,0} - \varphi_1$;
- $\delta_1^{(2)} := i^{(2)} - w_{1,2} + w_{1,1}$, $\delta_2^{(0)} := i^{(0)} - w_{2,0} + w_{2,2}$;
- $\delta_2^{(1)} := i^{(1)} - w_{2,1} + w_{2,0}$, $\delta_2^{(2)} := i^{(2)} - w_{2,2} + w_{2,1} - \varphi_2$.

It is straightforward that the distribution of $\{\delta_1^{(j)}, \delta_2^{(j)}\}_{j \in \mathbb{Z}_3}$ are uniformly random under the condition $\delta_1 := \sum_{j=0}^{2} \delta_1^{(j)} = i - \varphi_1$ and $\delta_2 := \sum_{j=0}^{2} \delta_2^{(j)} = i - \varphi_2$. Whereas in the hybrid $\mathcal{H}_2$, we have

- $\delta_1^{(0)} := i^{(0)} - w_{1,0} + w_{1,2}$, $\delta_1^{(1)} := -w_{1,1} + w_{1,0} - \varphi_1$;

- $\delta_1^{(2)} := -w_{1,2} + w_{1,1}$, $\delta_2^{(0)} := i^{(0)} - w_{2,0} + w_{2,2}$;
- $\delta_2^{(1)} := -w_{2,1} + w_{2,0}$, $\delta_2^{(2)} := -w_{2,2} + w_{2,1} - \varphi_2$.

The distribution of modified $\{\delta_1^{(j)}, \delta_2^{(j)}\}_{j\in\mathbb{Z}_3}$ are uniformly random under the condition $\delta_1 := i^{(0)} - \varphi_1$ and $\delta_2 := i^{(0)} - \varphi_2$. Note that the opened δ_1, δ_2 are also uniformly random in both hybrid $\mathcal{H}_1$ and hybrid $\mathcal{H}_2$ because of the random φ_1, φ_2. Using the opened δ_1, δ_2 and the input i from the environment $\mathcal{Z}$, the adversary $\mathcal{A}$ can extract $\tilde{\varphi}_1 := i - \delta_1$, $\tilde{\varphi}_2 := i - \delta_2$. The adversary $\mathcal{A}$ can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_1$ if and only if $\mathcal{A}$ can distinguish whether $\mathcal{K}_{\varphi_1}^{(1)}$ is generated by $\mathsf{DPF.Gen}(1^\lambda, f_{\tilde{\varphi}_1,1})$ or whether $\mathcal{K}_{\varphi_2}^{(0)}$ is generated by $\mathsf{DPF.Gen}(1^\lambda, f_{\tilde{\varphi}_2,1})$ without the knowledge of $\mathcal{K}_{\varphi_1}^{(0)}, \mathcal{K}_{\varphi_2}^{(1)}$. In other words, if there exists an adversary $\mathcal{A}$ who can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_1$ then we can construct an adversary $\mathcal{B}$ who uses $\mathcal{A}$ in a blackbox fashion can break either of the two $\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}} := (\mathsf{Gen}, \mathsf{Eval})$ above. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\epsilon_2 := 2 \cdot \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$. □

The adversary's view of $\mathcal{H}_2$ is identical to the simulated view $\mathsf{Exec}_{\mathcal{F}_{\mathsf{sot}}^{N,\ell},\mathcal{S},\mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda, \mathcal{A}) + 2 \cdot \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \ .$$

This concludes the proof. □

B Proof of Theorem 2

Theorem 2. Let $\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}$ be a secure function secret sharing scheme for the offset interval containment function $f_{p,q,r^{\mathsf{in}}}^{\mathsf{IC}}(x) : \mathbb{Z}_{2^{\ell_1}} \mapsto \mathbb{Z}_{2^{\ell_0}}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_0}}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$. The protocol $\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}$ as described in Fig. 6 UC-realizes $\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1}$ as described in Fig. 5 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

$$\begin{aligned} &6 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) + 9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) \\ &+ 2 \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) \ . \end{aligned}$$

Proof. To prove Thm. 2, we construct a PPT simulator Sim such that no non-uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\mathsf{Exec}_{\Pi_{\mathsf{csot}}^{\ell_0,\ell_1},\mathcal{A},\mathcal{Z}}$ where the parties $\mathcal{S} := \{S_0, S_1, S_2\}$ run protocol $\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}$ in the real world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\mathsf{Exec}_{\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1},\mathsf{Sim},\mathcal{Z}}$ where the parties S_0, S_1, S_2 interact with functionality $\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1}$ in the ideal world, and corrupted parties are controlled by the simulator Sim. Since the protocol is symmetric, we assume S_0 is corrupted for readability.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Since the semi-honest setting, Sim can obtain the correct input $\mathbf{x}^{(0)}, \mathbf{m}^{(0)}$ by simulating a dummy corrupted party S_0 to receive the messages from the environment $\mathcal{Z}$ in the ideal world. Sim simulates the interface of honest parties S_1, S_2. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon initialization, the simulator Sim acts as the honest party $S_j, j \in \{1,2\}$ to do:
  - Set $\rho_j \leftarrow \mathbb{Z}_{2^{\ell_1}}$, $\tau := 2^{\ell_1 - 1} - 1$;
  - Set $(\mathcal{K}_{\rho_j}^{(0)}, \mathcal{K}_{\rho_j}^{(1)}) \leftarrow \mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\rho_j,0}^{\mathrm{IC}})$ for the function $f_{0,\tau,\rho_j,0}^{\mathrm{IC}} : \mathbb{Z}_{2^{\ell_1}} \to \mathbb{Z}_{2^{\ell_0}}$;
  - Send $(\mathsf{sid}, \mathcal{K}_{\rho_j}^{(0)})$ to S_{j+1}, $(\mathsf{sid}, \mathcal{K}_{\rho_j}^{(1)})$ to S_{j+2}.
- The simulator Sim picks random $w_{k,j} \leftarrow \mathbb{Z}_{2^{\ell_1}}$ and $\zeta_{j,i} \leftarrow \mathbb{Z}_{2^{\ell_0}}$ for $k, j \in \mathbb{Z}_3, i \in \mathbb{Z}_2$;
- Upon receiving $(\textsc{Sel}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1,2\}$ from the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell_0}$, Sim does:
  - For $k \in \mathbb{Z}_3$, set $\delta_k^{(j)} := -w_{k,j} + w_{k,j+2} \pmod{2^{\ell_1}}$;
  - Set $\delta_j^{(j)} := \delta_j^{(j)} + \rho_j \pmod{2^{\ell_1}}$;
  - For $i \in \mathbb{Z}_2$, set $\tilde{x}_i^{(j)} := \zeta_{j,i} - \zeta_{j+2,i} \pmod{2^{\ell_0}}$

– Send $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+1}^{(j)}, \tilde{\mathbf{x}}^{(j)})$ to S_{j+2}, send $(\mathsf{sid}, \delta_j^{(j)}, \delta_{j+2}^{(j)})$ to S_{j+1} on behalf of the honest party S_j;

- Upon receiving $(\mathsf{sid}, \delta_0^{(0)}, \delta_1^{(0)}, \tilde{\mathbf{x}}^{(0)})$ from the corrupted S_0 to S_2 and $(\mathsf{sid}, \delta_0^{(0)}, \delta_2^{(0)})$ from the corrupted S_0 to S_1, the simulator Sim does:
  – Send $(\textsc{Sel}, \mathsf{sid}, \mathbf{x}^{(0)}, \mathbf{m}^{(0)})$ to the external $\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1}$;
  – For $k \in \{1,2\}$, set $\delta_k := \delta_k^{(\cdot)} + \delta_k^{(1)} + \delta_k^{(2)} \pmod{2^{\ell_1}}$;
  – Set $\beta_1^{(1)} \leftarrow \mathsf{DICF.Eval}_{0,\tau}^{\mathsf{IC}}(1, \mathcal{K}_{\rho_1}^{(1)}, \delta_1)$;
  – Set $\beta_2^{(0)} \leftarrow \mathsf{DICF.Eval}_{0,\tau}^{\mathsf{IC}}(0, \mathcal{K}_{\rho_2}^{(0)}, \delta_2)$;
  – Set $y^{(0)} := \tilde{x}_1^{(0)} + \sum_{q=0}^{1}\sum_{k=0}^{1}(-1)^k \cdot \tilde{x}_k^{(q)} \cdot \beta_{1+q}^{(1-q)} + \mathsf{PRF}_{\eta_0}^{\mathbb{Z}_{2^{\ell_0}}}(\mathsf{sid}, 2) - \mathsf{PRF}_{\eta_2}^{\mathbb{Z}_{2^{\ell_0}}}(\mathsf{sid}, 2) \pmod{2^{\ell_0}}$;
  – Send $(\textsc{Rand}, \mathsf{sid}, y^{(0)})$ to the external $\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1}$.

Indistinguishability. We assume that the parties S_0, S_1, S_2 communicate with each other via the secure channel functionality $\mathcal{F}_{\mathsf{sc}}$ (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0, \ldots, \mathcal{H}_3$.

Hybrid $\mathcal{H}_0$: It is the real protocol execution $\mathsf{Exec}_{\Pi_{\mathsf{csot}}^{\ell_0,\ell_1}, \mathcal{A}, \mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w_{k,j}\}_{k,j\in\mathbb{Z}_3}$ are picked uniformly random from $\mathbb{Z}_{2^{\ell_1}}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}$; $\{\zeta_{j,i}\}_{j\in\mathbb{Z}_3, i\in\mathbb{Z}_2}$ are picked uniformly random from $\mathbb{Z}_{2^{\ell_0}}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}$.

Claim 3. *If $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_0}}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$, $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_1$ and $\mathcal{H}_0$ are indistinguishable with advantage $\epsilon_1 := 6 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) + 9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$.*

Proof. We have changed 6 $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}$ outputs and 9 $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}$ outputs to uniformly random strings; therefore, the overall advantage is $6 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) + 9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$ by hybrid argument via reduction. □

Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_2$:

- For $j \in \{1,2\}$, the simulator Sim does:
  – Generate $\rho_j \leftarrow \mathbb{Z}_{2^{\ell_1}}$;
  – For $k \in \mathbb{Z}_3$, set $\delta_k^{(j)} := -w_{k,j} + w_{k,j+2} \pmod{2^{\ell_1}}$;
  – Set $\delta_j^{(j)} := \delta_j^{(j)} + \rho_j \pmod{2^{\ell_1}}$;

instead of

- For $j \in \{1,2\}$, the honest party S_j does:
  – Generate $\rho_j \leftarrow \mathbb{Z}_{2^{\ell_1}}$;
  – For $k \in \mathbb{Z}_3$, set $\delta_k^{(j)} := m_1^{(j)} - m_0^{(j)} - w_{k,j} + w_{k,j+2} \pmod{2^{\ell_1}}$;
  – Set $\delta_j^{(j)} := \delta_j^{(j)} + \rho_j \pmod{2^{\ell_1}}$;

Claim 4. *If $\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}}, \mathbb{Z}_{2^{\ell_0}}}$ be a secure function secret sharing scheme for the offset interval containment function $f_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}^{\mathsf{IC}}(x) : \mathbb{Z}_{2^{\ell_1}} \mapsto \mathbb{Z}_{2^{\ell_0}}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}}, \mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\epsilon_2 := 2 \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}}, \mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$.*

Proof. Note that ρ_1 and ρ_2 are used to generate the DICF keys $\mathcal{K}_{\rho_1}^{(0)}, \mathcal{K}_{\rho_1}^{(1)} \leftarrow \mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\rho_1,0}^{\mathrm{IC}})$ and $\mathcal{K}_{\rho_2}^{(0)}, \mathcal{K}_{\rho_2}^{(1)} \leftarrow \mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\rho_2,0}^{\mathrm{IC}})$, and the corrupted party S_0 only sees $\mathcal{K}_{\rho_1}^{(1)}, \mathcal{K}_{\rho_2}^{(0)}, \{\delta_1^{(j)}, \delta_2^{(j)}\}_{j\in\mathbb{Z}_3}$; therefore, the modification of $\mathcal{K}_{\rho_1}^{(0)}, \mathcal{K}_{\rho_2}^{(1)}, \{\delta_0^{(j)}\}_{j\in\mathbb{Z}_3}$ is oblivious to S_0. In the hybrid $\mathcal{H}_1$, we have

- $\delta_1^{(0)} := m_1^{(0)} - m_0^{(0)} - w_{1,0} + w_{1,2} \pmod{2^{\ell_1}}$;
- $\delta_1^{(1)} := m_1^{(1)} - m_0^{(1)} - w_{1,1} + w_{1,0} + \rho_1 \pmod{2^{\ell_1}}$;
- $\delta_1^{(2)} := m_1^{(2)} - m_0^{(2)} - w_{1,2} + w_{1,1} \pmod{2^{\ell_1}}$;
- $\delta_2^{(0)} := m_1^{(0)} - m_0^{(0)} - w_{2,0} + w_{2,2} \pmod{2^{\ell_1}}$;
- $\delta_2^{(1)} := m_1^{(1)} - m_0^{(1)} - w_{2,1} + w_{2,0} \pmod{2^{\ell_1}}$;
- $\delta_2^{(2)} := m_1^{(2)} - m_0^{(2)} - w_{2,2} + w_{2,1} + \rho_2 \pmod{2^{\ell_1}}$.

It is straightforward that the distribution of $\{\delta_1^{(j)}, \delta_2^{(j)}\}_{j\in\mathbb{Z}_3}$ are uniformly random under the condition $\delta_1 := \sum_{k=0}^{2} \delta_1^{(k)} = m_1 - m_0 + \rho_1$ and $\delta_2 := \sum_{k=0}^{2} \delta_2^{(k)} = m_1 - m_0 + \rho_2$. Whereas $\delta_1 := m_1^{(0)} - m_0^{(0)} + \rho_1$ and $\delta_2 := m_1^{(0)} - m_0^{(0)} + \rho_2$ in the hybrid $\mathcal{H}_2$. Using the opened δ_1, δ_2 and the input m_0, m_1 from the environment $\mathcal{Z}$, the adversity $\mathcal{A}$ can extract $\tilde{\rho}_1 := \delta_1 - m_1 + m_0, \tilde{\rho}_2 := \delta_2 - m_1 + m_0$. The adversary $\mathcal{A}$ can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_1$ if and only if $\mathcal{A}$ can distinguish whether $\mathcal{K}_{\rho_1}^{(1)}$ is generated by $\mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\tilde{\rho}_1,0}^{\mathrm{IC}})$ or whether $\mathcal{K}_{\rho_2}^{(0)}$ is generated by $\mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\tilde{\rho}_2,0}^{\mathrm{IC}})$ without the knowledge of $\mathcal{K}_{\rho_1}^{(0)}, \mathcal{K}_{\rho_2}^{(1)}$. In other word, if there exists an adversary $\mathcal{A}$ who can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_1$ then we can construct an adversary $\mathcal{B}$ who uses $\mathcal{A}$ in a blackbox fashion can break either of the two $\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}} := (\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ above. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\epsilon_2 := 2 \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A})$. □

Hybrid $\mathcal{H}_3$: $\mathcal{H}_3$ is the same as $\mathcal{H}_2$ except that in $\mathcal{H}_2$:
- For $j \in \{1,2\}, i \in \mathbb{Z}_2$, $\tilde{x}_i^{(j)} := \zeta_{j,i} - \zeta_{j+2,i} \pmod{2^{\ell_0}}$;

instead of
- For $j \in \{1,2\}, i \in \mathbb{Z}_2$, $\tilde{x}_i^{(j)} := x_i^{(j)} + \zeta_{j,i} - \zeta_{j+2,i} \pmod{2^{\ell_0}}$.

Claim 5. *$\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable.*

Proof. Since $\{\zeta_{j,i}\}_{j\in\mathbb{Z}_3, i\in\mathbb{Z}_2}$ are uniformly random in $\mathbb{Z}_{2^{\ell_0}}$, the distribution of $\{\tilde{x}_0^{(j)}, \tilde{x}_1^{(j)}\}_{j\in\mathbb{Z}_3}$ in $\mathcal{H}_2$ and $\mathcal{H}_3$ are identical. Therefore, $\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable. □

The adversary's view of $\mathcal{H}_3$ is identical to the simulated view $\mathsf{Exec}_{\mathcal{F}_{\mathsf{csot}}^{\ell_0,\ell_1},\mathcal{S},\mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$6 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) + 9 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 2 \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_0}}}}(1^\lambda, \mathcal{A}) \ .$$

This concludes the proof. □

C Proof of Theorem 3

Theorem 3. Let $\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}$ be a secure function secret sharing scheme for $f_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}^{\mathsf{IC}}(x) : \mathbb{Z}_{2^\ell} \mapsto \mathbb{Z}_{2^\lambda}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}$ be a secure function secret sharing scheme for point function $f_{\alpha,\beta}(x) : \mathbb{Z}_{\widetilde{m}_c+1} \mapsto \mathbb{Z}_{2^\ell}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\ell}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_n} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_n$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\lambda}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{\widetilde{m}_c+1}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A})$. The protocols $\Pi_{\mathsf{os}}^{\mathsf{const}}$ as described in Fig. 9 and $\Pi_{\mathsf{eval}}^{\mathsf{const}}$ as described in Fig. 10 UC-realize $\mathcal{F}_{\mathsf{bp}}^3$ as described in Fig. 2 in the $\mathcal{F}_{\mathsf{sot}}$-hybrid model against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage at most

$$\begin{aligned} &3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + 3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A}) \\ &+ \widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) + \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \\ &+ \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A}) + 2(\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \\ &+ (\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) \ . \end{aligned}$$

Proof. To prove Thm. 3, we construct a PPT simulator Sim such that no non-uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\mathsf{Exec}_{\{\Pi_{\mathsf{os}}^{\mathsf{const}},\Pi_{\mathsf{eval}}^{\mathsf{const}}\},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{sot}}}$ where the parties M, D, $\mathcal{S} := \{S_0, S_1, S_2\}$ run protocol $\Pi_{\mathsf{os}}^{\mathsf{const}}$, $\Pi_{\mathsf{eval}}^{\mathsf{const}}$ in the $\mathcal{F}_{\mathsf{sot}}$-hybrid world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\mathsf{Exec}_{\mathcal{F}_{\mathsf{bp}}^3,\mathsf{Sim},\mathcal{Z}}$ where the parties M, D, S_0, S_1, S_2 interact with functionality $\mathcal{F}_{\mathsf{bp}}^3$ in the ideal world, and corrupted parties are controlled by the simulator Sim. We consider following cases.

Case 1: S_0 (or S_1) is corrupted.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of $\mathcal{F}_{\mathsf{sot}}$ as well as honest parties M, D, S_1, S_2. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon receiving $(\textsc{Model}, \mathsf{sid}, M, (m, d))$ from the external $\mathcal{F}_{\mathsf{bp}}^3$, the simulator Sim computes $\widetilde{m}_c := 2^{d-1} - 1$ and acts as the honest model owner M to do:
  - **For** $i := 0$ to $\widetilde{m}_c - 1$:
    - Pick random $k_i^{(0)} \leftarrow \mathbb{Z}_n$, $t_i^{(0)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - Set $P_i^{(0)} := \{k_i^{(0)}, t_i^{(0)}\}$;
  - **For** $i := 0$ to $\widetilde{m}_c$, pick random $v_i^{(0)} \leftarrow \mathbb{Z}_{2^\ell}$;
  - Send $(\mathcal{P}^{(0)}, \mathbf{v}^{(0)})$ to S_0.
- Upon receiving $(\textsc{Data}, \mathsf{sid}, D, n)$ from the external $\mathcal{F}_{\mathsf{bp}}^3$, the simulator Sim acts as the honest data owner D to do:
  - **For** $i \in \mathbb{Z}_n$, pick random $x_i^{(0)}, x_i^{(1)}, x_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$;
  - Send $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ to S_0.
- Upon initialization, the simulator Sim acts as the honest party S_2 to do:
  - **For** $\tau := 2^{\ell-1} - 1$, $i := 0$ to $\widetilde{m}_c - 1$:
    - Generate $\rho_i \leftarrow \mathbb{Z}_{2^\ell}$;
    - Set $\mathcal{K}_{\rho_i}^{(0)}, \mathcal{K}_{\rho_i}^{(1)} \leftarrow \mathsf{DICF.Gen}_{0,\tau}^{\mathsf{IC}}(1^\lambda, f_{0,\tau,\rho_i,0}^{\mathsf{IC}})$ for the function $f_{0,\tau,\rho_i,0}^{\mathsf{IC}} : \mathbb{Z}_{2^\ell} \to \mathbb{Z}_{2^\lambda}$;
  - Send $(\mathsf{sid}, \mathcal{K}_{\rho_i}^{(0)})_{i \in \mathbb{Z}_{\widetilde{m}_c}}$ to S_0, $(\mathsf{sid}, \mathcal{K}_{\rho_i}^{(1)})_{i \in \mathbb{Z}_{\widetilde{m}_c}}$ to S_1;
- The simulator Sim picks random $w_{i,j} \leftarrow \mathbb{Z}_{2^\ell}$, $w'_{i,j} \leftarrow \mathbb{Z}_n$ for $i \in \mathbb{Z}_{\widetilde{m}_c}, j \in \mathbb{Z}_3$;
- Upon receiving $(\mathsf{Eval}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1, 2\}$ from the external $\mathcal{F}_{\mathsf{bp}}^3$, Sim does:
  - **For** $i := 0$ to $\widetilde{m}_c - 1$, set:
    - Set $k_i^{(j)} := w'_{i,j} - w'_{i,j+2}$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{x}^{(j+1)}, k_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$;
    - $\Delta x_i^{(j)} := (j-1) \cdot (\rho_i - t_i^{(j)} + x_{k_i}^{(j)}) + w_{i,j} - w_{i,j+2}$;
  - Send $(\mathsf{sid}, \Delta\mathbf{x}^{(j)})$ to S_0 on behalf of the honest party S_j.
- When the simulated $\mathcal{F}_{\mathsf{sot}}$ receives input from the corrupted party S_0, Sim sends $(\mathsf{Eval}, \mathsf{sid})$ to the external $\mathcal{F}_{\mathsf{bp}}^3$;
- Upon receiving $(\mathsf{sid}, \hat{\mathbf{c}}^{(0)}, \hat{\mathbf{v}}^{(0)})$ from the corrupted party S_0, the simulator Sim send $(\mathsf{Eval}, \mathsf{sid})$ to the external $\mathcal{F}_{\mathsf{bp}}^3$ and acts as the honest party S_2 to do:
  - Pick random $p \leftarrow \mathbb{Z}_{\widetilde{m}_c+1}$;
  - Set $(\mathcal{K}_p^{(0)}, \mathcal{K}_p^{(1)}) \leftarrow \mathsf{DPF.Gen}(1^\lambda, f_{p,1})$ for the point function $f_{p,1} : \mathbb{Z}_{\widetilde{m}_c+1} \to \mathbb{Z}_{2^\ell}$;
  - Send $(\mathsf{sid}, \mathcal{K}_p^{(0)})$ to S_0 on behalf of the honest party S_2;
- When the simulated receiver R terminates, the simulator Sim allows the $(\textsc{Result}, \mathsf{sid}, y)$ message to be delivered to R in the ideal world.

Indistinguishability. We assume that the parties S_0, S_1, S_2 communicate with each other via the secure channel functionality $\mathcal{F}_{\mathsf{sc}}$ (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0, \ldots, \mathcal{H}_3$.

Hybrid $\mathcal{H}_0$: It is the real execution $\mathsf{Exec}^{\mathcal{F}_{\mathsf{sot}}}_{\{\Pi_{\mathsf{os}}^{\mathsf{const}}, \Pi_{\mathsf{eval}}^{\mathsf{const}}\}, \mathcal{A}, \mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w_{i,j}\}_{i \in \mathbb{Z}_{\widetilde{m}_c}, j \in \mathbb{Z}_3}$ are picked uniformly random from $\mathbb{Z}_{2^\ell}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}$, and $\{w'_{i,j}\}_{i \in \mathbb{Z}_{\widetilde{m}_c}, j \in \mathbb{Z}_3}$ are picked uniformly random from $\mathbb{Z}_n$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_n}$

Claim 6. *If* $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\ell}$ *is a secure pseudorandom function with adversarial advantage* $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$*, and* $\mathsf{PRF}^{\mathbb{Z}_n} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_n$ *is a secure pseudorandom function with adversarial advantage* $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A})$*, then* $\mathcal{H}_1$ *and* $\mathcal{H}_0$ *are indistinguishable with advantage* $\epsilon_1 := 3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + 3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A})$.

Proof. We have changed $3\widetilde{m}_c$ $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}$ outputs and $3\widetilde{m}_c$ $\mathsf{PRF}^{\mathbb{Z}_n}$ outputs to uniformly random strings; therefore, the overall advantage is $3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + 3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A})$ by hybrid argument via reduction. □

Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_2$:

- **For** $i := 0$ to $\widetilde{m}_c - 1$:
  - Set $\Delta x_i^{(1)} := w_{i,1} - w_{i,0} \pmod{2^\ell}$;
  - Set $\Delta x_i^{(2)} := \rho_i - t_i^{(0)} + x_{k_i}^{(0)} + w_{i,2} - w_{i,1} \pmod{2^\ell}$;

instead of

- **For** $i := 0$ to $\widetilde{m}_c - 1$:
  - Set $\Delta x_i^{(1)} := t_i^{(1)} - x_{k_i}^{(1)} + w_{i,1} - w_{i,0} \pmod{2^\ell}$;
  - Set $\Delta x_i^{(2)} := \rho_i - x_{k_i}^{(2)} + w_{i,2} - w_{i,1} \pmod{2^\ell}$;

Claim 7. *If* $\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}} := (\mathsf{Gen}, \mathsf{Eval})$ *is a secure function secret sharing scheme for the offset interval containment function* $f^{\mathsf{IC}}_{p,q,r^{\mathsf{in}},r^{\mathsf{out}}}(x) : \mathbb{Z}_{2^\ell} \mapsto \mathbb{Z}_{2^\lambda}$ *with adversarial advantage* $\mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$*, then* $\mathcal{H}_2$ *and* $\mathcal{H}_1$ *are indistinguishable with advantage* $\epsilon_2 := \widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$.

Proof. For $i \in \mathbb{Z}_{\widetilde{m}_c}$, in the hybrid $\mathcal{H}_1$, it is straightforward that the distribution of $\{\Delta x_i^{(j)}\}_{j\in\mathbb{Z}_3}$ are uniformly random under the condition $\Delta x_i := \sum_{j=0}^{2} \Delta x_i^{(j)} = t_i - x_{k_i} + \rho_i$, where ρ_i is used to generate the DICF keys $\mathcal{K}^{(0)}_{\rho_i}, \mathcal{K}^{(1)}_{\rho_i} \leftarrow \mathsf{DICF}.\mathsf{Gen}^{\mathsf{IC}}_{0,\tau}(1^\lambda, f^{\mathsf{IC}}_{0,\tau,\rho_i,0})$. Whereas $\Delta x_i := t_i^{(0)} + \rho_i$ in the hybrid $\mathcal{H}_2$, we can show that if there exists an adversary $\mathcal{A}$ who can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_1$ then we can construct an adversary $\mathcal{B}$ who uses $\mathcal{A}$ in a blackbox fashion can break $\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}} := (\mathsf{Gen}, \mathsf{Eval})$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\epsilon_2 := \widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$. □

Hybrid $\mathcal{H}_3$: $\mathcal{H}_3$ is the same as $\mathcal{H}_2$ except that in $\mathcal{H}_3$:

- Pick random $p \leftarrow \mathbb{Z}_{\widetilde{m}_c+1}$;

instead of

- Pick p under the condition $\hat{c}_p^{(1)} + \hat{c}_p^{(2)} = 0 \pmod{2^\lambda}$.

Claim 8. *If* $\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}$ *be a secure function secret sharing scheme for point function* $f_{\alpha,\beta}(x) : \mathbb{Z}_{\widetilde{m}_c+1} \mapsto \mathbb{Z}_{2^\ell}$ *with adversarial advantage* $\mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$*, then* $\mathcal{H}_2$ *and* $\mathcal{H}_3$ *are indistinguishable with advantage* $\epsilon_2 := \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$.

Proof. p is used to generate the DPF keys $(\mathcal{K}_p^{(0)}, \mathcal{K}_p^{(1)}) \leftarrow \mathsf{DPF}.\mathsf{Gen}(1^\lambda, f_{p,1})$ for the point function $f_{p,1} : \mathbb{Z}_{\widetilde{m}_c+1} \to \mathbb{Z}_{2^\ell}$. We can show that if there exists an adversary $\mathcal{A}$ who can distinguish the view of $\mathcal{H}_2$ from the view of $\mathcal{H}_1$ then we can construct an adversary $\mathcal{B}$ who uses $\mathcal{A}$ in a blackbox fashion can break $\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}} := (\mathsf{Gen}, \mathsf{Eval})$ with the same advantage. Therefore, $\mathcal{H}_3$ and $\mathcal{H}_2$ are indistinguishable with adversarial advantage $\epsilon_1 := \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$. □

The adversary's view of $\mathcal{H}_3$ is identical to the simulated view $\mathsf{Exec}_{\mathcal{F}^3_{\mathsf{bp}},\mathcal{S},\mathcal{Z}}$. Therefore, the overall distinguishing advantage of case 1 is

$$3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + 3\widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_n}}(1^\lambda, \mathcal{A}) + \widetilde{m}_c \cdot \mathsf{Adv}_{\mathsf{DICF}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) + \mathsf{Adv}_{\mathsf{DPF}^{\mathbb{Z}_{\widetilde{m}_c+1},\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) \ .$$

Case 2: S_2 is corrupted.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of $\mathcal{F}_{\mathsf{sot}}$ as well as honest parties M, D, S_0, S_1. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon receiving $(\textsc{Model}, \mathsf{sid}, M, (m, d))$ from the external $\mathcal{F}^3_{\mathsf{bp}}$, the simulator Sim computes $\widetilde{m}_c := 2^{d-1} - 1$;
- Upon receiving $(\textsc{Data}, \mathsf{sid}, D, n)$ from the external $\mathcal{F}^3_{\mathsf{bp}}$, the simulator Sim acts as the honest data owner D to do:
  - **For** $i \in \mathbb{Z}_n$, pick random $x_i^{(0)}, x_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$;
  - Send $(\mathbf{x}^{(0)}, \mathbf{x}^{(2)})$ to S_2.
- Upon receiving $(\mathsf{Eval}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{0, 1\}$ from the external $\mathcal{F}^3_{\mathsf{bp}}$, Sim does:
  - **For** $i := 0$ to $\widetilde{m}_c - 1$, send $(\textsc{Fetch}, \mathsf{sid}, (0,0), (0,0), 0)$ to $\mathcal{F}^{n,\ell}_{\mathsf{sot}}$;

- When the simulated $\mathcal{F}_{\mathsf{sot}}$ receives input from the corrupted party S_2, Sim sends $(\mathsf{Eval}, \mathsf{sid})$ to the external $\mathcal{F}^3_{\mathsf{bp}}$;
- Upon receiving $(\mathsf{sid}, \Delta\mathbf{x}^{(2)})$ from the corrupted S_2 to S_0 and S_1, the simulator Sim does:
  - Pick random $p \leftarrow \mathbb{Z}_{\widetilde{m}_c+1}$;
  - **For** $i := 0$ to $\widetilde{m}_c$:
    - Pick random $c_i \leftarrow \mathbb{Z}^*_{2^\lambda}$, $c_i^{(0)} \leftarrow \mathbb{Z}_{2^\lambda}$;
    - **If** $i = p$, set $c_i = 0$;
    - Set $c_i^{(1)} := c_i - c_i^{(0)} \pmod{2^\lambda}$;
    - Pick random $w_i^{(0)} \leftarrow \mathbb{Z}_{2^\ell}$, $w_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - **For** $j \in \{0, 1\}$, set $\hat{c}_i^{(j)} := c_i^{(j)}$, $\hat{v}^{(j)} := -w_i^{(j)}$;
  - Send $(\mathsf{sid}, \hat{\mathbf{c}}^{(0)}, \hat{\mathbf{v}}^{(0)})$ to S_2 on behalf of the party S_0;
  - Send $(\mathsf{sid}, \hat{\mathbf{c}}^{(1)}, \hat{\mathbf{v}}^{(1)})$ to S_2 on behalf of the party S_1;
- When the simulated receiver R terminates, the simulator Sim allows the $(\textsc{Result}, \mathsf{sid}, y)$ message to be delivered to R in the ideal world.

Indistinguishability. We assume that the parties S_0, S_1, S_2 communicate with each other via the secure channel functionality $\mathcal{F}_{\mathsf{sc}}$ (omitted in the protocol description for simplicity). The indistinguishability is proven through a series of hybrid worlds $\mathcal{H}_0, \mathcal{H}_1, \mathcal{H}_2$.
Hybrid $\mathcal{H}_0$: It is the real execution $\mathsf{Exec}^{\mathcal{F}_{\mathsf{sot}}}_{\{\Pi^{\mathsf{const}}_{\mathsf{os}}, \Pi^{\mathsf{const}}_{\mathsf{eval}}\}, \mathcal{A}, \mathcal{Z}}$.
Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, δ is picked uniformly random from $\mathbb{Z}_{\widetilde{m}_c+1}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}$, and $\{w_i^{(0)}, w_i^{(1)}\}_{i\in\mathbb{Z}_{\widetilde{m}_c+1}}$ are picked uniformly random from $\mathbb{Z}_{2^\ell}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}$.

Claim 9. *If $\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{\widetilde{m}_c+1}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A})$, and $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\ell}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_1$ and $\mathcal{H}_0$ are indistinguishable with advantage $\epsilon_1 := \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A}) + 2(\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$.*

Proof. We have changed 1 $\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}$ outputs and $2(\widetilde{m}_c + 1)$ $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}$ outputs to uniformly random strings; therefore, the overall advantage is $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A}) + 2(\widetilde{m}_c+1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$ by hybrid argument via reduction. □

Hybrid $\mathcal{H}_2$: $\mathcal{H}_2$ is the same as $\mathcal{H}_1$ except that in $\mathcal{H}_2$, $\{c_i\}_{i\in\mathbb{Z}_{\widetilde{m}_c+1}}$ are picked uniformly random from $\mathbb{Z}_{2^\lambda}$ instead of calculating from the outputs of $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}$.

Claim 10. *If $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\lambda}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\epsilon_2 := (\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$.*

Proof. In the hybrid $\mathcal{H}_1$, $\{c_i\}_{i\in\mathbb{Z}_{\widetilde{m}_c+1}}$ are calculated by summing up the share of edge costs (i.e., outputs of $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}$ r_i) along i-th leaf node's path. It is straightforward that each c_i is the sum of the elements of a unique subset of $\{r_i\}_{i\in\mathbb{Z}_{\widetilde{m}_c}}$. Whereas $\{c_i\}_{i\in\mathbb{Z}_{\widetilde{m}_c+1}}$ are uniformly random strings in the hybrid $\mathcal{H}_2$, they can be separated out uniformly random strings $\{r_i\}_{i\in\mathbb{Z}_{\widetilde{m}_c}}$ by subtracting adjacent two elements. Therefore, we have equivalently changed $\widetilde{m}_c + 1$ $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}$ outputs, and the overall advantage is $\epsilon_2 := (\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$ by hybrid argument via reduction. □

The adversary's view of $\mathcal{H}_2$ is identical to the simulated view $\mathsf{Exec}_{\mathcal{F}^3_{\mathsf{bp}}, \mathcal{S}, \mathcal{Z}}$. Therefore, the overall distinguishing advantage of case 2 is

$$\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{\widetilde{m}_c+1}}}(1^\lambda, \mathcal{A}) + 2(\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + (\widetilde{m}_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) \ .$$

This concludes the proof. □

D Proof of Theorem 4

Theorem 4. The protocol $\Pi_{\mathsf{os}}^{\mathsf{linear}}$ as described in Fig. 13 and $\Pi_{\mathsf{eval}}^{\mathsf{linear}}$ as described in Fig. 14 UC-realizes $\mathcal{F}_{\mathsf{bp}}^3$ as described in Fig. 2 in the $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$-hybrid model against semi-honest adversaries who can statically corrupted up to 1 server.

Proof. To prove Thm. 4, we construct a PPT simulator Sim such that no non-uniform PPT environment $\mathcal{Z}$ can distinguish between (i) the real execution $\mathsf{Exec}^{\mathcal{F}_{\mathsf{sot}},\mathcal{F}_{\mathsf{csot}}}_{\{\Pi_{\mathsf{os}}^{\mathsf{linear}},\Pi_{\mathsf{eval}}^{\mathsf{linear}}\},\mathcal{A},\mathcal{Z}}$ where the parties M, D, $\mathcal{S} := \{S_0, S_1, S_2\}$ run protocol $\Pi_{\mathsf{os}}^{\mathsf{linear}}$, $\Pi_{\mathsf{eval}}^{\mathsf{linear}}$ in the $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$-hybrid world and the corrupted parties are controlled by a dummy adversary $\mathcal{A}$ who simply forwards messages from/to $\mathcal{Z}$, and (ii) the ideal execution $\mathsf{Exec}_{\mathcal{F}_{\mathsf{bp}}^3,\mathsf{Sim},\mathcal{Z}}$ where the parties M, D, S_0, S_1, S_2 interact with $\mathcal{F}_{\mathsf{bp}}^3$ in the ideal world, and corrupted parties are controlled by Sim. Since the protocol is symmetric, we assume S_0 is corrupted for readability.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$ as well as honest parties M, D, S_1, S_2. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon receiving $(\textsc{Model}, \mathsf{sid}, M, (m, d))$ from the external $\mathcal{F}_{\mathsf{bp}}^3$, Sim acts as the honest model owner M to do:
  - **For** $i \in \mathbb{Z}_m, j \in \mathbb{Z}_3$, pick random:
    - $\mathcal{I}_i^{\mathsf{left},(j)}, \mathcal{I}_i^{\mathsf{right},(j)} \leftarrow \mathbb{Z}_m$, $\mathcal{J}_i^{\mathsf{left},(j)}, \mathcal{J}_i^{\mathsf{right},(j)} \leftarrow \mathbb{Z}_n$;
    - $t_i^{(j)}, v_i^{(j)} \leftarrow \mathbb{Z}_{2^\ell}$;
  - Pick random $\mathsf{id}_1^{(0)} \leftarrow \mathbb{Z}_m$, $k_1^{(0)} \leftarrow \mathbb{Z}_n$;
  - Send $(\mathcal{P}^{(0)}, \mathsf{id}_1^{(0)}, k_1^{(0)})$ to S_0;
- Upon receiving $(\textsc{Data}, \mathsf{sid}, D, n)$ from the external $\mathcal{F}_{\mathsf{bp}}^3$, the simulator Sim acts as the honest data owner D to do:
  - **For** $i := 0$ to $n-1$, pick random $x_i^{(0)}, x_i^{(1)}, x_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$;
  - Send $\mathbf{x}^{(0)}$ to S_0.
- Upon receiving $(\mathsf{Eval}, \mathsf{sid}, S_j)$ for an honest party S_j, from the external $\mathcal{F}_{\mathsf{bp}}^3$, Sim does:
  - **For** $i := 1$ to d :
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{x}^{(j+1)}, 0)$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathcal{P}^{(j)}, \mathcal{P}^{(j+1)}, 0)$ to $\mathcal{F}_{\mathsf{sot}}^{\widetilde{m},*}$;
    - Pick random $y^{(j)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - **If** $i \geq d$, return $y^{(j)}$ to the receiver R and **break**;
    - Send $(\textsc{Sel}, \mathsf{sid}, (0, 0), (0, 0))$ to $\mathcal{F}_{\mathsf{csot}}^{\log \widetilde{m},\ell}$;
    - Send $(\textsc{Sel}, \mathsf{sid}, (0, 0), (0, 0))$ to $\mathcal{F}_{\mathsf{csot}}^{\log n,\ell}$;
- When the simulated $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$ receives input from the corrupted party S_j, the simulator Sim sends $(\mathsf{Eval}, \mathsf{sid})$ to the external $\mathcal{F}_{\mathsf{bp}}^3$;
- When the simulated receiver R terminates, the simulator Sim allows the $(\textsc{Result}, \mathsf{sid}, y)$ message to be delivered to R in the ideal world.

Indistinguishability. We assume that the parties M, D, S_0, S_1, S_2 communicate with each other via the secure channel functionality $\mathcal{F}_{\mathsf{sc}}$ (omitted in the protocol description for simplicity). The views of $\mathcal{A}$ and $\mathcal{Z}$ in $\mathsf{Exec}^{\mathcal{F}_{\mathsf{sot}},\mathcal{F}_{\mathsf{csot}}}_{\{\Pi_{\mathsf{os}}^{\mathsf{linear}},\Pi_{\mathsf{eval}}^{\mathsf{linear}}\},\mathcal{A},\mathcal{Z}}$ and $\mathsf{Exec}_{\mathcal{F}_{\mathsf{bp}}^3,\mathsf{Sim},\mathcal{Z}}$ are identical. Therefore, it is perfectly indistinguishable.

This concludes the proof. □