UC Secure Private Branching Program and Decision Tree Evaluation

Keyu Ji
jikeyu@zju.edu.cn
Zhejiang University

Bingsheng Zhang
bingsheng@zju.edu.cn
Zhejiang University

Tianpei Lu
21921147@zju.edu.cn
Zhejiang University

Lichun Li
lichun.llc@antgroup.com
Ant Group

Kui Ren
kuiren@zju.edu.cn
Zhejiang University

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 multi-party computation platform for outsourced branching program and/or decision tree evaluation. We construct a constant-round protocol and a poly-round protocol. In particular, the overall (online + offline) communication cost of our poly-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 poly-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 [2, 3, 7, 15, 16, 18, 21, 22, 26]. These works can be divided into two main categories based on protocol round complexity: (i) constant-round solutions [3, 7, 15, 21, 26], and (ii) polynomial-round solutions [11, 16, 20, 22]. As summaries in [18], 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, polynomial-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) [22], which can be realized by either OT or ORAM. The OT-based OAI private decision tree evaluation protocol proposed in [22] takes linear communication (in tree size, m) and $4d$ rounds. When OAI is realized by Circuit ORAM [24], the online communication complexity can be reduced to $O(d^4)$, but it takes up to $O(d^2)$ rounds. Moreover, the ORAM initialization phase is very slow for large tree size and/or feature numbers. For instance, it could take 20 days to insert 2^{16} elements of 512 bits each [14].

The best polynomial-round solution is recently proposed by Ma *et al.* [20]. 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 linear communication in the offline phase. Meanwhile, the protocol proposed by [20] 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 [20] 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. We first propose a new 1-out-of-N shared OT protocol with logarithmic communication. In a shared OT, given a vector of shared messages $\mathbf{x} := (x_0, \ldots, x_{N-1})$ and an shared index $i \in \mathbb{Z}_N$, the MPC parties can jointly obtain x_i in the shared form without revealing i. Our approach follows the line of research initiated by Boyle *et al*. [5], 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*. [12] adopt DPF in the MPC setting to achieve ORAM. In [12], both servers S_1 and S_2 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_1 and S_2 first generates the DPF keys for $f_i(x)$ via MPC. After obtaining the shared $\widetilde{x}_i$, S_1 and S_2 then needs to obliviously evaluate $\mathsf{PRF}_k(i)$ via MPC to decrypt x_i. Therefore, the entire process is time-consuming.

In this work, we eliminate the needs of aforementioned two costly MPC operations by introducing more servers. Our platform utilizes four servers $S_1, \ldots, S_4$. Suppose S_1 and S_2 needs to evaluate the DPF on i, where i is additively shared among four servers. To avoid MPC generation of DPF keys, we let a third server (an non-evaluator of this DPF), say S_3, generate a pair of DPF keys $(\mathcal{K}^{(1)}, \mathcal{K}^{(2)})$ on $f_\varphi(x)$ for a random $\varphi \in \mathbb{Z}_N$ in the offline phase. S_3 then sends $\mathcal{K}^{(1)}$ and $\mathcal{K}^{(2)}$ to S_1 and S_2, respectively. In online phase, $\delta := i - \varphi \pmod N$ is opened to the evaluators, i.e. S_1 and S_2. The evaluators then cyclic-shift the messages vector $\mathbf{x}$ to the left δ positions and evaluate DPF $f_\varphi(x)$ on the shifted messages to obtain shared x_i. To avoid oblivious PRF evaluation via MPC, we share the messages $\mathbf{x}$ among four servers using replicated additively shares. That is, S_j, $j \in [4]$ holds share $\mathbf{x}^{(j)} := (x_k^{(j)})_{k \in \mathbb{Z}_N}$, where $x_k^{(1)} = x_k^{(2)}$, $x_k^{(3)} = x_k^{(4)}$, and $x_k = x_k^{(1)} + x_k^{(3)} = x_k^{(2)} + x_k^{(4)}$. During the shared OT, S_1 and S_2 evaluate DPF on the same shares of messages $\mathbf{x}^{(1)}$(or $\mathbf{x}^{(2)}$) and obtain $x_i^{(1)}$(or $x_i^{(2)}$) in the additively shared form. Meanwhile, S_3 and S_4 evaluate the other DPF on the same shares of messages $\mathbf{x}^{(3)}$(or $\mathbf{x}^{(4)}$) and obtain $x_i^{(3)}$(or $x_i^{(4)}$) in the additively shared form. Then they re-randomize shares of x_i to ensure the uniform distribution in local. Namely, x_i is additively shared among the four servers.

Our constant-round solution. We construct a 3-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 four 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 comparison function* (DCF) [4]. More specifically, S_4 plays the role of DCF key generator while S_1 and S_2 play the role of DCF evaluators. In the offline phase, S_4 precomputes the DCF keys and distribute them to S_1 and S_2. In the online phase, the servers mask the difference of its threshold and feature, and open it to S_1 and S_2. They then jointly evaluate DCF to securely compare the corresponding feature with the threshold. When the feature is less than the threshold, S_1 and S_2 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_1 and S_2 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. After that, S_1 and S_2 open the shifted path costs and re-share the shifted classification values to S_3 and S_4. They output the classification value of the leaf node whose path cost is 0 as the evaluation result to the receiver.

Our polynomial-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 proposed shared OT protocols 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 DCF 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 DCF to perform secure comparison between the corresponding threshold and feature in a 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.

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 for 40-bit security), λ_3 is the size of DGK ciphertext (= 2048 for 40-bit security), λ_4 is the size of Paillier ciphertext (= 4096 for 40-bit security), 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		
[2] (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	○
[21] (HHH)	-	$((m_c+n)\ell+2(m_c+1)+n)\lambda_2$	4	○
[18] (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	○
[18] (HGH)	$5m_c\ell\lambda_1$	$(m_c+n)\lambda_3+m_c\ell(2\lambda_1+1)+(3m_c+2)\lambda_2$	6	○
[1]**	-	$n\ell\lambda_1+\lambda_5+\lambda_6+(2N+5)\lambda_7$	2	●
Ours (constant)	$4\cdot 2^d(\log n+\log\ell+d)\lambda_8$	$2^d(3\log n+3d+7\ell)$	3	●
[11]	$6(2^dn\ell+d(3\ell-\log l-2)+2^d-1)$	$4(2^dn\ell+d(3\ell-\log l-2)+2^d-1)$	$\log\ell+d+1$	●
[27]	$6((2^dn+4)\ell-5)$	$4((2^dn+4)\ell-5)$	$2\ell-1$	●
[19]**	$6(2^d-1)\ell$	$3\cdot 2^{d-1}\lambda_4+4(2^d-1)\ell$	$d+1$	●
[22] (OT)	$6d\ell\lambda_1$	$d((m+n)\ell+2(\log m+\log n)\lambda_1)$	$4d$	○
[20] (complete)	$2^d(\ell+\log n)$	$d(4\lambda_1+n\ell+(7\ell+8)\lambda_1)$	$2d-1$	●
[20] (sparse)	$m(\ell+\log n+\lambda_1+3d)$	$d((4\lambda_1+n\ell)+(7\ell+8)\lambda_1+8)$	$2d-1$	○
Ours (poly-round)	$4d(\log n+\log m+\log\ell)\lambda_8$	$12d(2(\log m+\log n)+\ell)$	$2d-1$	●

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

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 •. 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 $\lambda_1,\dots,\lambda_8$ to differentiate them. For instance, λ_4 refers to the ciphertext size of Paillier encryption, which is 4096 bits for 40-bit security; whereas, the security parameter λ_8 is the 128-bit AES key size in our schemes. Note that some works (marked with **), e.g., [1,19] 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. 8) With regards to polynomial-round solutions, [20] 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 polynomial-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 $[\![x]\!] := \{x^{(1)},x^{(2)}\}$, where $x^{(1)}+x^{(2)}=x \pmod n$. Denote a $(4,4)$-additive secret sharing in $\mathbb{Z}_n$ by $\langle x\rangle := \{x^{(1)},x^{(2)},x^{(3)},x^{(4)}\}$, where $x^{(1)}+x^{(2)}+x^{(3)}+x^{(4)}=x \pmod n$. 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 $x\in[-2^{\ell-1},2^{\ell-1}]$ to $\mathbb{Z}_{2^\ell}$, i.e.,when x is negative, $x'=x+2^{\ell-1}$.

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.* [5]. Given a function family $\mathcal{F} = \{f : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}\}$, a dealer uses the FSS scheme for $\mathcal{F}$ to split a function $f \in \mathcal{F}$ into two additive shares f_1, f_2, such that for $\forall x \in \mathbb{G}^{in}$, $f_1(x) + f_2(x) = f(x) \pmod{|\mathbb{G}^{\mathsf{out}}|}$.

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

- $\mathsf{Gen}(1^\lambda, \alpha, \beta, \mathbb{G}^{\mathsf{in}}, \mathbb{G}^{\mathsf{out}})$ is a key generation algorithm that outputs a pair of keys $(\mathcal{K}^{(1)}, \mathcal{K}^{(2)})$. 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. For $\forall x \in \mathbb{G}^{\mathsf{in}}$, $\forall b \in \{1,2\}$, it outputs $\beta_x^{(b)} \in \mathbb{G}^{\mathsf{out}}$, such that $\beta_x^{(1)} + \beta_x^{(2)} = f_{\alpha,\beta}(x) \pmod{|\mathbb{G}^{\mathsf{out}}|}$.

When DPF is used to realize a PIR protocol, the servers need to run Eval on every element of the input domain. Boyle *et al.* [6] provide 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 $f^<_{\alpha,\beta} : \mathbb{G}^{\mathsf{in}} \to \mathbb{G}^{\mathsf{out}}$, outputting β if $0 \le x < \alpha$ and 0 if $x \ge \alpha$. To enable secure comparison on private input, the recent work of Boyle et al. [4] provides an FSS schemes $(\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ for the class of *offset* comparison functions $\mathcal{F}^{\mathsf{IC}}$ with given offset r^{in} and r^{out}, such that for $\forall f^{\mathsf{IC}} \in \mathcal{F}^{\mathsf{IC}}$, $f^{\mathsf{IC}}_{\alpha,\beta}(x + r^{\mathsf{in}}) - r^{\mathsf{out}} = f^<_{\alpha,\beta}(x)$. We focus on the case of $r^{\mathsf{out}} = 0$. Similar to DPF, it consists of a pair of algorithms $(\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ as follows:

- $\mathsf{Gen}^{\mathsf{IC}}(1^\lambda, \alpha, \beta, \mathbb{G}^{\mathsf{in}}, \mathbb{G}^{\mathsf{out}}, r^{\mathsf{in}})$ generates $(\mathcal{K}^{(1)}, \mathcal{K}^{(2)})$.
- $\mathsf{Eval}^{\mathsf{IC}}(b, \mathcal{K}^{(b)}, \Delta x)$ outputs $c^{(b)} \in \mathbb{Z}_2$ for given $\Delta x := x + r^{\mathsf{in}}$, such that $c^{(1)} + c^{(2)} = f^<_{\alpha,\beta}(x)$.

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*

$$\mathsf{Adv}(1^\lambda, \mathcal{A}) = \left| \Pr\left[\begin{array}{l} (f_0, f_1, \phi) \leftarrow \mathcal{A}(1^\lambda); b \leftarrow \{0,1\}; \\ (\mathcal{K}^{(1)}, \mathcal{K}^{(2)}) \leftarrow \mathsf{Gen}(1^\lambda, f_b); \\ b^* \leftarrow \mathcal{A}((\mathcal{K}^{(i)})_{i \in T}, \phi) : \\ f_0, f_1 \in \mathcal{F} \ \wedge \ b = b^* \end{array} \right] - \frac{1}{2} \right|$$

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 four non-colluding computing servers $\mathcal{S} := \{S_1, \ldots, S_4\}$, 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.

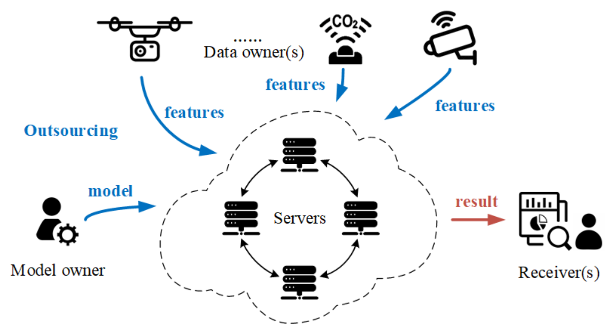


Figure 1: System architecture.

Universal Composability. Our security model is based on the Universal Composibility (UC) framework [8], 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

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

It interacts with a set of computing servers $\mathcal{S} := \{S_1, \ldots, S_\kappa\}$, the model owner M, the data owner D, the receiver R, 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 $(\text{MODEL}, \mathsf{sid}, \mathcal{M})$ from the model owner M:
  - Send notification $(\text{MODEL}, \mathsf{sid}, M, (\mathcal{M}.m, \mathcal{M}.d))$ to Sim;
  - Set $\mathsf{status} := 1$;
  - Record $\mathcal{M}$;
- Upon receiving $(\text{DATA}, \mathsf{sid}, \mathbf{x})$ from the data owner D, if $\mathsf{status} = 1$:
  - Send notification $(\text{DATA}, \mathsf{sid}, D, |\mathbf{x}|)$ to Sim;
  - Set $\mathsf{status} := 2$;
  - Record $\mathbf{x}$;

Evaluation phase:

- Upon receiving $(\text{EVAL}, \mathsf{sid})$ 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 $(\text{RESULT}, \mathsf{sid}, y)$ to R via input delayed channel;

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

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 $\mathcal{S} := \{S_1, \ldots, S_\kappa\}$, the model owner M, the data owner D, and the receiver R only communicate with an ideal functionality $\mathcal{F}_{\mathsf{bp}}^{\kappa}$ 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 $(\text{EVAL}, \mathsf{sid})$ to the functionality $\mathcal{F}_{\mathsf{bp}}^{\kappa}$, $\mathcal{F}_{\mathsf{bp}}^{\kappa}$ runs $y \leftarrow \mathcal{M}(\mathbf{x})$ and then sends $(\text{RESULT}, \mathsf{sid}, y)$ 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 $\mathcal{S} := \{S_1, \ldots, S_\kappa\}$ 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:*

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

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

It interacts with $\mathcal{S} := \{S_1, \ldots, S_4\}$ and the adversary Sim.

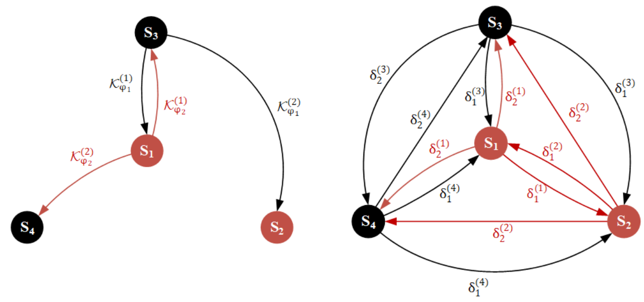
- Upon receiving $(\text{FETCH}, \mathsf{sid}, \mathbf{x}^{(j)} := (x_0^{(j)}, \ldots, x_{N-1}^{(j)}), i^{(j)})$ from $S_j \in \mathcal{S}$:
  - Send notification $(\text{FETCH}, \mathsf{sid}, S_j)$ to Sim;
  - Record $(\mathbf{x}^{(j)}, i^{(j)})$;
- Once all players have submitted their input, does:
  - Assert $\mathbf{x}^{(1)} = \mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)} = \mathbf{x}^{(4)}$;
  - Compute $i := \sum_{j=1}^{4} i^{(j)} \pmod N$;
  - Upon receiving $(\text{RAND}, \mathsf{sid}, y^*)$ from Sim for the corrupted party S_k:
    - Pick random $y^{(1)}, \ldots, y^{(4)} \in \mathbb{Z}_{2^\ell}$ under the constraint $\sum_{j=1}^{4} y^{(j)} = x_i^{(1)} + x_i^{(3)} \pmod{2^\ell}$ and $y^{(k)} = y^*$;
    - Send $(\text{RETURN}, \mathsf{sid}, y^{(j)})$ to all parties $S_j \in \mathcal{S}$ via private delayed channel.

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

4 OT with logarithmic communication

In the 1-out-of-N shared OT protocol, given a vector of shared messages $\mathbf{x} := (x_0, \ldots, x_{N-1})$ and an shared index $i \in \mathbb{Z}_N$, the MPC parties can jointly obtain x_i in the shared form without revealing i. We propose an efficient shared OT protocol with logarithmic communication in both offline and online phase. As depicted in Fig. 3, our shared OT is a 4-party computation protocol. The messages are replicated shared, while the index is additively shared. Let $\mathbf{x}^{(j)} := (x_0^{(j)}, \ldots, x_{N-1}^{(j)})$ and $i^{(j)}$ be the shares of player S_j, $j \in [4]$. We have $\mathbf{x}^{(1)} = \mathbf{x}^{(2)}$ and $\mathbf{x}^{(3)} = \mathbf{x}^{(4)}$; messages $\mathbf{x} = \mathbf{x}^{(1)} + \mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \mathbf{x}^{(4)}$ and $i = \sum_{j=1}^{4} i^{(j)} \pmod N$. To facilitate our private decision tree and BP evaluation protocol, the output of shared OT is additively shared among the four players; nevertheless, it is possible to add 1 round share conversion to any other shared type at the end.

Intuition. Our construction utilizes the DPF technique [5] in a novel way, and in this work the output of DPF for $f_i(x)$ is additive shared in $\mathbb{Z}_{2^\ell}$ instead of $\mathsf{GF}(2^\ell)$. Conventionally, in a DPF-based two-server OT protocol, the client holds an index $i \in \mathbb{Z}_N$ and both servers hold the messages $\mathbf{x} \in (\mathbb{Z}_{2^\ell})^N$. During the protocol, the client generates a pair of DPF keys $(\mathcal{K}^{(1)}, \mathcal{K}^{(2)})$ for $f_i(x)$ and then distributes them to the two servers. The servers then jointly evaluate and return shares of $x_i := \sum_{j=0}^{N-1} f_i(j) \cdot x_j$ to the client. On the other hand, in shared OT, the index and messages are both stored in the shared form. To address the former issue, we let a third server (an non-evaluator of this DPF), say S_3, generate a pair of



(a) Offline. (b) Online.

Figure 4: Communication diagram for protocol $\Pi_{\mathsf{sot}}^{N,\ell}$.

DPF keys $(\mathcal{K}^{(1)}, \mathcal{K}^{(2)})$ on $f_\varphi(x)$ for a random $\varphi \in \mathbb{Z}_N$ in the offline phase. S_3 then sends $\mathcal{K}_1$ and $\mathcal{K}_2$ to S_1 and S_2, respectively. In online phase, $\delta := i - \varphi \pmod N$ is opened to the evaluators, i.e. S_1 and S_2. The evaluators then cyclic-shift the messages vector $\mathbf{x}$ to the left δ positions and evaluate DPF $f_\varphi(x)$ on the shifted messages to obtain shared x_i. To address the latter issue, the messages are replicated shared, i.e., $\mathbf{x} := \mathbf{x}^{(1)} + \mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \mathbf{x}^{(4)}$, such that S_1 and S_2 (S_3 and S_4) hold the same share; therefore, they can perform DPF evaluation on the shares instead of the plaintext.

Protocol description. Our 1-round shared OT protocol is designed in the online/offline model (cf. Fig. 5). During the initialization, S_1 and S_3 agree on a random seed $\eta_1 \in \{0,1\}^\lambda$; S_2 and S_4 agree on a random seed $\eta_2 \in \{0,1\}^\lambda$; S_1 and S_2 agree on a random seed $\eta_3 \in \{0,1\}^\lambda$; S_3 and S_4 agree on a random seed $\eta_4 \in \{0,1\}^\lambda$. In offline phase, S_3 and S_1 act as DPF generator locally invoke $\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}.\mathsf{Gen}$ with random input $\varphi_1, \varphi_2 \leftarrow \mathbb{Z}_N$ to get DPF keys $(\mathcal{K}_{\varphi_1}^{(1)}, \mathcal{K}_{\varphi_1}^{(2)})$ and $(\mathcal{K}_{\varphi_2}^{(1)}, \mathcal{K}_{\varphi_2}^{(2)})$, respectively. Then S_3 sends $\mathcal{K}_{\varphi_1}^{(1)}$ to S_1, $\mathcal{K}_{\varphi_1}^{(2)}$ to S_2; S_1 sends $\mathcal{K}_{\varphi_2}^{(1)}$ to S_3, $\mathcal{K}_{\varphi_2}^{(2)}$ to S_2. In online phase, four servers compute $\langle\delta_1\rangle := \langle i\rangle - [\![\varphi_1]\!]$ and $\langle\delta_2\rangle_N := \langle i\rangle - [\![\varphi_2]\!]$ with fresh random mask w_1 and w_2. Then reveal δ_1 to S_1 and S_2, δ_2 to S_3 and S_4, as shown in Fig. 4b. For $j \in \{1,2\}$, S_j first cyclic-shifts the share of messages $\mathbf{x}^{(j)}$ to the left δ_1 positions and denotes the array after shift as $\widetilde{\mathbf{x}}^{(j)}$. Next, S_j invokes $\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}.\mathsf{EvalAll}$ with the DPF key that received in offline phase. After that, S_1 and S_2 obtain a secret shared array $(\beta_{k,\varphi_1})_{k\in\mathbb{Z}_N}$ whose the only non-zero value is $\beta_{\varphi_1,\varphi_1} = 1$. We have $x_i = \sum_{j=1}^{4} \hat{y}^{(j)} \pmod{2^\ell}$ where

$$\hat{y}^{(j)} := \sum_{k=0}^{N-1} (\widetilde{x}_k^{(1)} \cdot \beta_{k,\varphi_1}^{(j)}) \pmod{2^\ell}.$$

Finally, we re-randomize $\hat{y}^{(j)}$ to ensure the uniform distribution.

Efficiency. $\Pi_{\mathsf{sot}}^{N,\ell}$ is a one-round 1-out-of-N shared OT protocol with offline communication cost $4\lambda \log N$ bits and online communication cost 12ℓ bits.

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

Initialization:

- S_1 and S_3 agree on a random seed $\eta_1 \leftarrow \{0,1\}^\lambda$;
- S_2 and S_4 agree on a random seed $\eta_2 \leftarrow \{0,1\}^\lambda$.
- S_1 and S_2 agree on a random seed $\eta_3 \leftarrow \{0,1\}^\lambda$;
- S_3 and S_4 agree on a random seed $\eta_4 \leftarrow \{0,1\}^\lambda$.

Offline phase:

- Upon initialization, S_1 does:
  - Generate $\varphi_2 \leftarrow \mathbb{Z}_N$ and set $\varphi_2^{(1)} := \varphi_2 - \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_N}(\mathsf{sid}, 0)$;
  - Set $\mathcal{K}_{\varphi_2}^{(1)}, \mathcal{K}_{\varphi_2}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \varphi_2, 1, \mathbb{Z}_N, \mathbb{Z}_{2^\ell})$;
  - Send $\mathcal{K}_{\varphi_2}^{(1)}$ to S_3, $\mathcal{K}_{\varphi_2}^{(2)}$ to S_4;
- Upon initialization, S_3 dose:
  - Generate $\varphi_1 \leftarrow \mathbb{Z}_N$ and set $\varphi_1^{(1)} := \varphi_1 - \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_N}(\mathsf{sid}, 0)$;
  - Set $\mathcal{K}_{\varphi_1}^{(1)}, \mathcal{K}_{\varphi_1}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \varphi_1, r_1, \mathbb{Z}_N, \mathbb{Z}_{2^\ell})$;
  - Send $\mathcal{K}_{\varphi_1}^{(1)}$ to S_1, $\mathcal{K}_{\varphi_1}^{(2)}$ to S_2;
- Upon initialization, S_2 sets $\varphi_2^{(2)} := \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_N}(\mathsf{sid}, 0)$;
- Upon initialization, S_4 sets $\varphi_1^{(2)} := \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_N}(\mathsf{sid}, 0)$;

Online phase:

- Upon receiving $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, i^{(j)})$ from the environment $\mathcal{Z}$, player $S_j, j \in \{1,2\}$ does:
  - Set $w_1^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_N}(\mathsf{sid}, 1)$, $w_2^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_N}(\mathsf{sid}, 2)$;
  - Set $\delta_1^{(j)} := i^{(j)} + w_1^{(j)} \pmod N$;
  - Set $\delta_2^{(j)} := i^{(j)} - \varphi_2^{(j)} + w_2^{(j)} \pmod N$;
  - Send $\delta_1^{(j)}$ to S_{3-j}, $\delta_2^{(j)}$ to S_3 and S_4;
- Upon receiving $\delta_1^{(3-j)}$ from S_{3-j}, $\delta_1^{(3)}$ from S_3 , and $\delta_1^{(4)}$ from S_4, player $S_j, j \in \{1,2\}$ does:
  - Set $\delta_1 := \delta_1^{(1)} + \delta_1^{(2)} + \delta_1^{(3)} + \delta_1^{(4)} \pmod N$;
  - Set $\widetilde{x}_k^{(j)} := x_{k-\delta_1 \pmod N}^{(j)}$, for $k \in \mathbb{Z}_N$;
  - Set $(\beta_{k,\varphi_1}^{(j)})_{k\in\mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(j, \mathcal{K}_{\varphi_1}^{(j)})$;
  - Set $\zeta_j \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$, $\zeta_3 \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$;
  - Return $y^{(j)} := \sum_{k=0}^{N-1} (\widetilde{x}_k^{(j)} \cdot \beta_{k,\varphi_1}^{(j)}) + \zeta_j + (-1)^j \cdot \zeta_3$.
- Upon receiving $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, i^{(j)})$ from the environment $\mathcal{Z}$, player $S_j, j \in \{3,4\}$ does:
  - Set $w_1^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_N}(\mathsf{sid}, 1)$, $w_2^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_N}(\mathsf{sid}, 2)$;
  - Set $\delta_1^{(j)} := i^{(j)} - \varphi_1^{(j-2)} - w_1^{(j)} \pmod N$;
  - Set $\delta_2^{(j)} := i^{(j)} - w_2^{(j)} \pmod N$;
  - Send $\delta_1^{(j)}$ to S_1 and S_2, $\delta_2^{(j)}$ to S_{7-j};
- Upon receiving $\delta_2^{(1)}$ from S_1, $\delta_2^{(2)}$ from S_2, and $\delta_2^{(7-j)}$ from S_{7-j}, player $S_j, j \in \{3,4\}$ does:
  - Set $\delta_2 := \delta_2^{(1)} + \delta_2^{(2)} + \delta_2^{(3)} + \delta_2^{(4)} \pmod N$;
  - Set $\widetilde{x}_k^{(j)} := x_{k-\delta_2 \pmod N}^{(j)}$, for $k \in \mathbb{Z}_N$;
  - Set $(\beta_{k,\varphi_2}^{(j)})_{k\in\mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(j-2, \mathcal{K}_{\varphi_2}^{(j-2)})$;
  - Set $\zeta_{j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$, $\zeta_4 \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid})$;
  - Return $y^{(j)} := \sum_{k=0}^{N-1} (\widetilde{x}_k^{(j)} \cdot \beta_{k,\varphi_2}^{(j-2)}) - \zeta_{j-2} + (-1)^j \cdot \zeta_4$.

Figure 5: 1-out-of-n shared OT protocol $\Pi_{\mathsf{sot}}^{N,\ell}$.

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 Appendix A.

Functionality $\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2}$

It interacts with $\mathcal{S} := \{S_1,\ldots,S_4\}$ and the adversary Sim.

- Upon receiving $(\textsc{CompFetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{m}^{(j)})$ from $S_j \in \mathcal{S}$:
  - Send notification $(\textsc{CompFetch}, \mathsf{sid}, S_j)$ to Sim;
  - Record $(\mathbf{x}^{(j)}, \mathbf{m}^{(j)})$;
- Once all players have submitted their input, does:
  - For $k \in \{0,1\}$, compute $x_k := \sum_{j=1}^4 x_k^{(j)} \pmod{2^{\ell_1}}$ and $m_k := \sum_{j=1}^4 m_k^{(j)} \pmod{2^{\ell_2}}$;
  - Set $b \leftarrow (m_0 < m_1)$;
  - Upon receiving $(\textsc{Rand}, \mathsf{sid}, y^*)$ from Sim for the corrupted party S_k:
    - Pick random $y^{(1)},\ldots,y^{(4)} \in \mathbb{Z}_{2^{\ell_1}}$ under the constraint $\sum_{j=1}^4 y^{(j)} = x_{1-b} \pmod{2^{\ell_1}}$ and $y^{(k)} = y^*$;
    - Send $(\textsc{Return}, \mathsf{sid}, y^{(j)})$ to all parties $S_j \in \mathcal{S}$ via private delayed channel.

Figure 6: The conditional shared OT functionality $\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2}$

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. 5 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*

$$3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda,\mathcal{A}) + \mathsf{Adv}_{\mathsf{DPF}}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}(1^\lambda,\mathcal{A}) \quad .$$

5 Conditional Shared OT

In the conditional shared OT protocol, given a vector of shared messages $\mathbf{x} := (x_0,x_1) \in (\mathbb{Z}_{2^{\ell_1}})^2$ and two shared keywords $\mathbf{m} := (m_0,m_1) \in (\mathbb{Z}_{2^{\ell_2}})^2$, 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. 6, our conditional shared OT is a 4-party computation protocol. The messages and keywords are additively shared among the 4 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, $j \in [4]$. We have $\mathbf{x} = \sum_{j=1}^4 \mathbf{x}^{(j)}$ and $\mathbf{m} = \sum_{j=1}^4 \mathbf{m}^{(j)}$.

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. We compress the round complexity to one. In our protocol, the servers are divided into two groups. $\Delta m := m_1 - m_0$ is opened to each groups with the corresponding DCF offset, while the $(4,4)$-addictive secret sharing messages are converted to replicated secret sharing, where the servers of each group have the same shares. The two groups then perform two DCF evaluations in a parallel. Subsequently, the oblivious selection can be computed locally

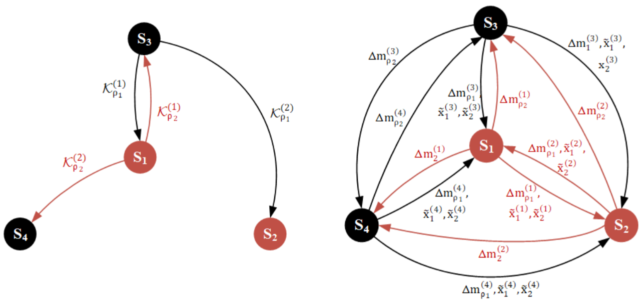
Protocol $\Pi_{\mathsf{csot}}^{\ell_1,\ell_2}$

Initialization:
- S_1 and S_3 agree on a random seed $\eta_1 \leftarrow \{0,1\}^\lambda$;
- S_2 and S_4 agree on a random seed $\eta_2 \leftarrow \{0,1\}^\lambda$;
- S_1 and S_2 agree on a random seed $\eta_3 \leftarrow \{0,1\}^\lambda$;
- S_3 and S_4 agree on a random seed $\eta_4 \leftarrow \{0,1\}^\lambda$.

Offline phase:
- Upon initialization, S_1 dose:
  - Generate $\rho_2 \leftarrow \mathbb{Z}_{2^{\ell_2}}$ and set $\rho_2^{(1)} := \rho_2 - \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},0)$;
  - $\mathcal{K}_{\rho_2}^{(1)}, \mathcal{K}_{\rho_2}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell_2-1}, 1, \mathbb{Z}_{2^{\ell_2}}, \mathbb{Z}_{2^{\ell_1}}, \rho_2)$;
  - Send $\mathcal{K}_{\rho_2}^{(1)}$ to S_3, $\mathcal{K}_{\rho_2}^{(2)}$ to S_4;
- Upon initialization, S_3 dose:
  - Generate $\rho_1 \leftarrow \mathbb{Z}_{2^{\ell_2}}$ and set $\rho_1^{(1)} := \rho_1 - \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},0)$;
  - $\mathcal{K}_{\rho_1}^{(1)}, \mathcal{K}_{\rho_1}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell_2-1}, 1, \mathbb{Z}_{2^{\ell_2}}, \mathbb{Z}_{2^{\ell_1}}, \rho_1)$;
  - Send $\mathcal{K}_{\rho_1}^{(1)}$ to S_1, $\mathcal{K}_{\rho_1}^{(2)}$ to S_2;
- Upon initialization, S_2 sets $\rho_2^{(2)} := \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},0)$;
- Upon initialization, S_4 sets $\rho_1^{(2)} := \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},0)$.

Online phase:
- Upon receiving $(\textsc{CompFetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{m}^{(j)})$ from the environment $\mathcal{Z}$, player $S_j, j \in \{1,2\}$ does:
  - Set $w_{m,1}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},1)$, $w_{m,2}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},2)$;
  - Set $w_{x,0}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},1)$, $w_{x,1}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},2)$;
  - Set $\Delta m_{\rho_1}^{(j)} := m_1^{(j)} - m_0^{(j)} + w_{m,1}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\Delta m_{\rho_2}^{(j)} := m_1^{(j)} - m_0^{(j)} + \rho_2^{(j)} + w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\tilde{x}_i^{(j)} := x_i^{(j)} + w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, for $i \in \{0,1\}$;
  - Send $(\Delta m_{\rho_1}^{(j)}, \tilde{\mathbf{x}}^{(j)})$ to S_{3-j}, $\Delta m_{\rho_2}^{(j)}$ to S_3 and S_4;
- Upon receiving $(\Delta m_{\rho_1}^{(3-j)}, \tilde{\mathbf{x}}^{(3-j)})$ from S_{3-j}, $(\Delta m_{\rho_1}^{(3)}, \tilde{\mathbf{x}}^{(3)})$ from S_3, $(\Delta m_{\rho_1}^{(4)}, \tilde{\mathbf{x}}^{(4)})$ from S_4, player $S_j, j \in \{1,2\}$ does:
  - Set $\Delta m_{\rho_1} := \sum_{i=1}^4 \Delta m_{\rho_1}^{(i)} \pmod{2^{\ell_2}}$;
  - Set $\hat{x}_0^{(j)} := \sum_{i=1}^4 \tilde{x}_0^{(i)} \pmod{2^{\ell_1}}$, $\hat{x}_1^{(j)} := \sum_{i=1}^4 \tilde{x}_1^{(i)} \pmod{2^{\ell_1}}$;
  - Set $\beta_1^{(j)} \leftarrow \mathsf{DCF.Eval}^{\mathsf{IC}}(j, \mathcal{K}_{\rho_1}^{(j)}, \Delta m_{\rho_1})$;
  - Set $\zeta_j \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},3)$, $\zeta_3 \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},1)$;
  - $y^{(j)} := \beta_1^{(j)} \cdot \hat{x}_0^{(j)} + (j-1-\beta_1^{(j)}) \cdot \hat{x}_1^{(j)} + \zeta_j + (-1)^j \cdot \zeta_3$;
- Upon receiving $(\textsc{CompFetch}, \mathsf{sid}, \mathbf{x}^{(j)}, \mathbf{m}^{(j)})$ from the environment $\mathcal{Z}$, player $S_j, j \in \{3,4\}$ does:
  - Set $w_{m,1}^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},1)$, $w_{m,2}^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^{\ell_2}}}(\mathsf{sid},2)$;
  - Set $w_{x,0}^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},1)$, $w_{x,1}^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},2)$;
  - Set $r_i \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},i)$, $i \in [4]$;
  - Set $\hat{x}_0^{(j)} := r_1 + r_2 \pmod{2^{\ell_1}}$, $\hat{x}_1^{(j)} := r_3 + r_4 \pmod{2^{\ell_1}}$;
  - Set $\Delta m_{\rho_1}^{(j)} := m_1^{(j)} - m_0^{(j)} + \rho_1^{(j-2)} - w_{m,1}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\Delta m_{\rho_2}^{(j)} := m_1^{(j)} - m_0^{(j)} - w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\tilde{x}_i^{(j)} := x_i^{(j)} - r_{j+2i-2} - w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, $i \in \{0,1\}$;
  - Send $\{\Delta m_{\rho_1}^{(j)}, \tilde{\mathbf{x}}^{(j)}\}$ to S_1 and S_2, $\Delta m_{\rho_2}^{(j)}$ to S_{7-j};
- Upon receiving $\Delta m_{\rho_2}^{(1)}$ from S_1, $\Delta m_{\rho_2}^{(2)}$ from S_2, $\Delta m_{\rho_2}^{(7-j)}$ from S_{7-j}, player $S_j, j \in \{3,4\}$ does:
  - $\Delta m_{\rho_2} := \sum_{i=1}^4 \Delta m_{\rho_2}^{(i)} \pmod{2^{\ell_2}}$;
  - $\beta_2^{(j-2)} \leftarrow \mathsf{DCF.Eval}^{\mathsf{IC}}(j-2, \mathcal{K}_{\rho_2}^{(j-2)}, \Delta m_{\rho_2})$;
  - Set $\zeta_{j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},3)$, $\zeta_4 \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid},5)$;
  - $y^{(j)} := \beta_2^{(j-2)} \cdot \hat{x}_0^{(j)} + (j-3-\beta_2^{(j-2)}) \cdot \hat{x}_1^{(j)} - \zeta_{j-2} + (-1)^j \cdot \zeta_4$.

Figure 7: Conditional shared OT Protocol $\Pi_{\mathsf{csot}}^{\ell_1,\ell_2}$.



(a) Offline.

(b) Online.

Figure 8: Communication diagram for protocol $\Pi_{\mathsf{csot}}^{\ell_1,\ell_2}$.

by scalar product.

Protocol description. Our 1-round conditional shared OT is depicted in Fig. 7. During the initialization, S_1 and S_3 agree on a random seed $\eta_1 \in \{0,1\}^\lambda$; S_2 and S_4 agree on a random seed $\eta_2 \in \{0,1\}^\lambda$; S_1 and S_2 agree on a random seed $\eta_3 \in \{0,1\}^\lambda$; S_3 and S_4 agree on a random seed $\eta_4 \in \{0,1\}^\lambda$.

In offline phase, S_1 invokes $\mathsf{DCF}^{2^{\ell_2},2^{\ell_1}}.\mathsf{Gen}^{\mathsf{IC}}$ with random offset ρ_2 and input $2^{\ell-1}$ (the decomposition point of positive and negative numbers) to generate $(\mathcal{K}_{\rho_2}^{(1)}, \mathcal{K}_{\rho_2}^{(2)})$, then distributes them to S_3 and S_4; S_3 invokes $\mathsf{DCF}^{2^{\ell_2},2^{\ell_1}}.\mathsf{Gen}^{\mathsf{IC}}$ with the other random offset ρ_1 and input $2^{\ell-1}$ to generate $(\mathcal{K}_{\rho_1}^{(1)}, \mathcal{K}_{\rho_1}^{(2)})$, then distributes them to S_1 and S_2.

In online phase, four servers reveal $\langle \Delta m_{\rho_1} \rangle := \langle m_1 \rangle - \langle m_0 \rangle + \rho_1$ to S_1 and S_2, reveal $\langle \Delta m_{\rho_2} \rangle := \langle m_1 \rangle - \langle m_0 \rangle + \rho_2$ to S_3 and S_4. Meanwhile, S_3 and S_4 generate $\hat{\mathbf{x}}^{(3)} := (\hat{x}_0^{(3)}, \hat{x}_1^{(3)})$ and $\hat{\mathbf{x}}^{(4)} := (\hat{x}_0^{(4)}, \hat{x}_1^{(4)})$ by PRF with the same random seed η_4 respectively, such that $\hat{\mathbf{x}}^{(3)} = \hat{\mathbf{x}}^{(4)}$; then four servers compute $\langle \tilde{\mathbf{x}} \rangle := \langle \mathbf{x} \rangle - [\![\hat{\mathbf{x}}^{(3)}]\!] \pmod{2^{\ell_1}}$ and reveal it to S_1 and S_2 with the help of fresh random mask w_1 and w_2. S_1 denotes $\tilde{\mathbf{x}}$ by $\hat{\mathbf{x}}^{(1)}$, S_2 denotes $\tilde{\mathbf{x}}$ by $\hat{\mathbf{x}}^{(2)}$, where messages $\mathbf{x} = \hat{\mathbf{x}}^{(1)} + \hat{\mathbf{x}}^{(3)} = \hat{\mathbf{x}}^{(2)} + \hat{\mathbf{x}}^{(4)}$ as shown in Fig. 8. After that, servers locally evaluate DCF with received key and masked input to obtain the shared comparison result β_1, β_2, then compute scalar multiplication and re-randomize the result to get the shares of selected message $\langle y \rangle$ in uniform distribution.

Efficiency. $\Pi_{\mathsf{csot}}^{\ell_1,\ell_2}$ is a one-round protocol with offline communication cost $4\lambda \log \ell_2$ bits and online communication cost $12(\ell_1 + \ell_2)$ bits.

Security. We show the security of our conditional shared OT Protocol $\Pi_{\mathsf{csot}}^{\ell_1,\ell_2}$ with the following theorem, and its proof can be found in Appendix B.

Theorem 2. *Let $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}$ be a secure function secret sharing scheme for offset comparison function $f_{\alpha,\beta}^{\mathsf{IC}}(x)$: $\mathbb{Z}_{2^{\ell_2}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A})$. 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 $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_2}}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A})$. The protocol $\Pi_{\mathsf{csot}}^{\ell_1,\ell_2}$ as described in Fig. 7 UC-realizes $\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2}$ as described in Fig. 6 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage*

$$8 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A}) + \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A}).$$

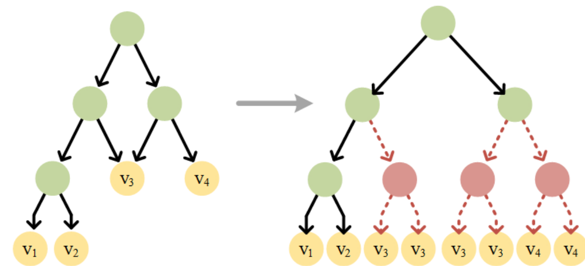


Figure 9: Complete tree depth-padding.

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

6.1 Constant-Round Protocol

Our constant-round protocol requires three communication rounds and a complete decision tree, which can be trans-

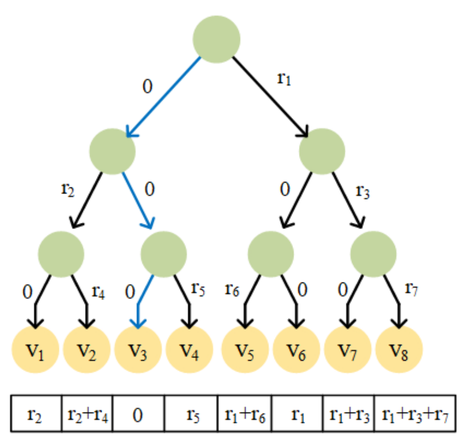


Figure 10: Path cost diagram.

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 $k_i^{(1)} \leftarrow \mathbb{Z}_n$, $k_i^{(2)} := k_i - k_i^{(1)} \pmod n$
    - Set $t_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$, $t_i^{(2)} := t_i - t_i^{(1)} \pmod{2^\ell}$;
    - Set $P_i^{(1)} := \{k_i^{(1)}, t_i^{(1)}\}$, $P_i^{(2)} := \{k_i^{(2)}, t_i^{(2)}\}$;
  - **foreach** element i in $\mathbf{v}$:
    - $v_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$, $v_i^{(2)} := v_i - v_i^{(1)} \pmod{2^\ell}$;
  - Send $(\mathcal{P}^{(1)}, \mathbf{v}^{(1)})$ to S_1, $(\mathcal{P}^{(2)}, \mathbf{v}^{(2)})$ to S_2.
- 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 $x_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$, set $x_i^{(2)} := x_i^{(1)}$;
    - Set $x_i^{(3)} := x_i - x_i^{(1)} \pmod{2^\ell}$, $x_i^{(4)} := x_i^{(3)}$;
  - Send $\mathbf{x}^{(j)}$ to S_j, $j \in [4]$.

Figure 11: Outsourcing Protocol $\Pi_{\mathsf{os}}^{\mathsf{const}}$.

formed from a normal DAG by adding dummy nodes as illustrated in Fig. 9, i.e. $\widetilde{m} = 2^d$, $\widetilde{m}_c = 2^{(d-1)} - 1$. 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_1 and S_2 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_1 and S_2 sum up the share of edge costs along all paths to get a vector of path costs for all leaf nodes in a shared form. As shown in Fig 10, 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. 11 to generate the additive share of $\mathcal{P}, \mathbf{v}$ and distribute them to S_1 and S_2. 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 four servers in replicated secret sharing. After the exection, for $j \in [4]$, S_j hold the shares $\mathbf{x}^{(j)} := (x_i^{(j)})_{i \in \mathbb{Z}_n}$, such that $\mathbf{x}^{(1)} = \mathbf{x}^{(2)}$, $\mathbf{x}^{(3)} = \mathbf{x}^{(4)}$ and $\mathbf{x} = \mathbf{x}^{(1)} + \mathbf{x}^{(3)} = \mathbf{x}^{(2)} + \mathbf{x}^{(4)}$.

Evaluation. Our constant-round protocol follows the modular design framework of [18]. As depicted in Fig. 12, 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_1 and S_2 , four servers construct the $(4,4)$-secret-sharing $\langle k_i \rangle$ by setting $k_i^{(3)} := 0$ and $k_i^{(3)} := 0$ in S_3 and S_4. Then four servers invoke our 1-out-of-N shared OT

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

Initialization:

- S_1 and S_3 agree on a random seed $\eta_1 \leftarrow \{0,1\}^\lambda$;
- S_2 and S_4 agree on a random seed $\eta_2 \leftarrow \{0,1\}^\lambda$;
- S_1 and S_2 agree on a random seed $\eta_3 \leftarrow \{0,1\}^\lambda$.

Offline phase:

- Upon initialization, S_3 dose:
  - **for** $i := 0$ to $\widetilde{m}_c - 1$:
    - Generate $\rho_i \leftarrow \mathbb{Z}_{2^\ell}$ and set $\rho_i^{(1)} := \rho_i - \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i)$;
    - $\mathcal{K}_{i,\rho}^{(1)}, \mathcal{K}_{i,\rho}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell-1}, 1, \mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}, \rho_i)$;
  - Send $(\mathcal{K}_{i,\rho}^{(1)})_{i \in \mathbb{Z}_{\widetilde{m}_c}}$ to S_1, $(\mathcal{K}_{i,\rho}^{(2)})_{i \in \mathbb{Z}_{\widetilde{m}_c}}$ to S_2;
- Upon initialization, S_4 sets $\rho_i^{(2)} := \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i)$, $i \in \mathbb{Z}_{\widetilde{m}_c}$.

Online phase:

- Upon receiving $(\mathsf{Eval}, \mathsf{sid})$ from the environment $\mathcal{Z}$, $S_j, j \in \{1,2\}$ does:
  - **for** $i := 0$ to $\widetilde{m}_c - 1$:
    - Set $\tilde{k}_i^{(j)} := k_i^{(j)}$, $w_i^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i)$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^j, \tilde{k}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    - Set $\Delta x_i^{(j)} := t_i^{(j)} - x_{k_i}^{(j)} + w_i^{(j)} \pmod{2^\ell}$;
  - Send $\Delta\mathbf{x}^{(j)}$ to S_{3-j};
- Upon receiving $\Delta\mathbf{x}^{(3-j)}$ from S_{3-j}, $\Delta\mathbf{x}^{(3)}$ from S_3 , and $\Delta\mathbf{x}^{(4)}$ from S_4, player $S_j, j \in \{1,2\}$ does:
  - **for** $i := 0$ to $\widetilde{m}_c - 1$:
    - Set $\Delta x_i := \sum_{q=1}^{4} \Delta x_i^{(q)} \pmod{2^\ell}$;
    - Set $b_i^{(j)} \leftarrow \mathsf{DCF.Eval}^{\mathsf{IC}}(j, \mathcal{K}_{i,\rho}^{(j)}, \Delta x_i)$;
    - Set $r_i \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^\lambda}}(\mathsf{sid}, i)$
    - Set $e_{i,1}^{(j)} := (j - 1 - b_i^{(j)}) \cdot r_i \pmod{2^\lambda}$;
    - Set $e_{i,2}^{(j)} := b_i^{(j)} \cdot r_i \pmod{2^\lambda}$;
  - Set $\delta \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{\widetilde{m}_c+1}}(\mathsf{sid}, 0)$;
  - **for** $i := 0$ to $\widetilde{m}_c$:
    - Sum up the share of edge costs along i-th leaf node's path to get $c_i^{(j)}$, set $\hat{c}_i^{(j)} := c_{i-\delta \pmod{m_c+1}}^{(j)}$;
    - Set $\hat{v}_{i,1}^{(j)} \leftarrow \mathbb{Z}_{2^\ell}$, $\hat{v}_{i,2}^{(j)} := v_{i-\delta \pmod{\widetilde{m}_c+1}}^{(j)} - \hat{v}_{i,1}^{(j)} \pmod{2^\ell}$;
  - Send $(\hat{c}_i^{(j)}, \hat{v}_{i,1}^{(j)})_{i \in \mathbb{Z}_{\widetilde{m}_c+1}}$ to S_3, $(\hat{c}_i^{(j)}, \hat{v}_{i,2}^{(j)})_{i \in \mathbb{Z}_{\widetilde{m}_c+1}}$ to S_4;
- Upon receiving $(\mathsf{Eval}, \mathsf{sid})$ from the environment $\mathcal{Z}$, $S_j, j \in \{3,4\}$ does:
  - **for** $i := 0$ to $\widetilde{m}_c - 1$:
    - Set $\tilde{k}_i^{(j)} := 0$, $w_i^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid}, i)$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^j, \tilde{k}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    - Set $\Delta x_i^{(j)} := \rho_i^{(j-2)} - x_{k_i}^{(j)} - w_i^{(j)} \pmod{2^\ell}$;
  - Send $\Delta\mathbf{x}^{(j)}$ to S_1 and S_2;
- Upon receiving $(\hat{c}_i^{(1)}, \hat{v}_{i,j-2}^{(1)})_{i \in \mathbb{Z}_{\widetilde{m}_c+1}}$ from S_1, $(\hat{c}_i^{(2)}, \hat{v}_{i,j-2}^{(2)})_{i \in \mathbb{Z}_{\widetilde{m}_c+1}}$ from S_2, player $S_j, j \in \{3,4\}$ does:
  - **for** $i := 0$ to $\widetilde{m}_c$:
    - $\hat{c}_i := \hat{c}_i^{(1)} + \hat{c}_i^{(2)} \pmod{2^\lambda}$;
    - **if** $\hat{c} = 0$ set $\hat{v}_i^{(j-2)} := \hat{v}_{i,j-2}^{(1)} + \hat{v}_{i,j-2}^{(2)} \pmod{2^\ell}$ and return $\hat{v}_i^{(j-2)}$ to the receiver R.

Figure 12: Constant-round Evaluation Protocol $\Pi_{\mathsf{eval}}^{\mathsf{const}}$ in the $\mathcal{F}_{\mathsf{sot}}$-hybrid model.

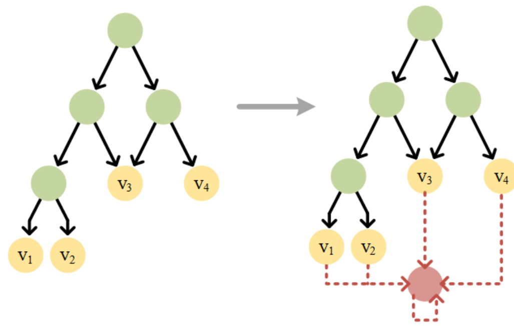


Figure 13: Sparse DAG depth-padding.

protocol described in Sec. 4 to obtain corresponding feature $\langle x_{k_i} \rangle$.

Comparison. Our comparison protocol is based on the DCF scheme [4], where S_4 plays the role of generator while S_1 and S_2 play the role of evaluator. To avoid leaking features and thresholds to servers, we let S_4 precompute the DCF keys for $\forall P_i \in \mathcal{P}$, which compares the input value with a random value ρ_i, and distribute the keys to evaluators S_1 and S_2. In online phase, four servers jointly compute $\Delta x_i := t_i - x_{k_i} + \rho_i$ and open it to DCF evaluators, S_1 and S_2. Then S_1 and S_2 are able to obtain shared comparison result vector $\{[\![b_i]\!]\}_{i \in \mathbb{Z}_{\widetilde{m}_c}}$, where $b_i := 1$ if $t_i - x_{k_i}$ is positive and $b_i := 0$ otherwise.

Path evaluation. S_1 and S_2 first generate random masks $(r_i)_{i \in \mathbb{Z}_{\widetilde{m}_c}}$ together. Next, for each decision node $P_i \in \mathcal{P}$, S_1 and S_2 locally compute its left out-going edge cost $[\![e_{\ell,i}]\!] := [\![(1-b_i) \cdot r_i]\!]$ and right out-going edge cost $[\![e_{r,i}]\!] := [\![b_i \cdot r_i]\!]$. For each leaf node $v_i \in \mathbf{v}$, S_1 and S_2 sum up the edge costs along its path from the root to get its path cost c_i in shared form. Subsequently, S_1 and S_2 generate a random number $\delta \leftarrow \mathbb{Z}_{\widetilde{m}_c+1}$ together. Denote path cost vector as $\mathcal{C}$ (cf. Fig. 10). S_1 and S_2 circular shift the $[\![\mathcal{C}]\!]$ and $[\![\mathbf{v}]\!]$ to the left δ positions. Then open $\mathcal{C}$ and reshare $\mathbf{v}$ to S_3 and S_4. S_3 and S_4 can easily select the classification value share v_i of evaluation result according to the position i of $c_i = 0$. Finally, they return the $v_i^{(1)}$ and $v_i^{(2)}$ to the receiver.

6.2 Polynomial-Round Protocol

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 Appendix C.

Theorem 3. *Let* $\mathsf{DPF}_{\mathsf{IC}}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}$ *be a secure function secret sharing scheme for offset comparison function* $f_{\alpha,\beta}(x) : \mathbb{Z}_{2^\ell} \mapsto \mathbb{Z}_{2^\lambda}$ *with adversarial advantage* $\mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}(1^\lambda, \mathcal{A})$*. 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* $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\lambda}$ *be a secure pseudorandom function with adversarial advantage* $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$*. The protocol*

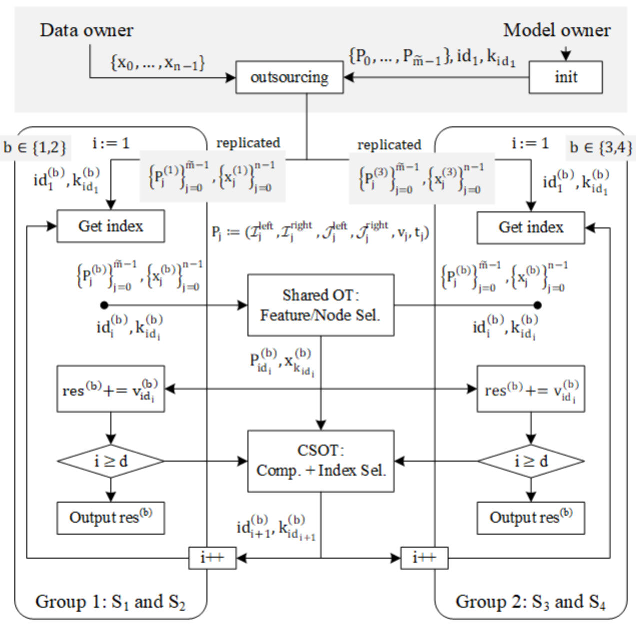


Figure 14: Overview of our polynomial-round protocol.

$\Pi_{\mathsf{os}}^{\mathsf{const}}$ *as described in Fig. 11 and* $\Pi_{\mathsf{eval}}^{\mathsf{const}}$ *as described in Fig. 12 UC-realizes* $\mathcal{F}_{\mathsf{bp}}^4$ *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*

$$3m_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + (m_c+1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) + m_c \cdot \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}(1^\lambda, \mathcal{A})$$

Our polynomial-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. 13. 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

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

- Upon receiving ($\textsc{Model}, \mathsf{sid}, \mathcal{P}$) from the environment $\mathcal{Z}$, the model owner M:
  - Build the position mapping, denote i-th element as $P_i := \{I_i^{\mathsf{left}}, I_i^{\mathsf{right}}, \mathcal{J}_i^{\mathsf{left}}, \mathcal{J}_i^{\mathsf{right}}, t_i, v_i\}$;
  - **for** $i := 0$ to $\widetilde{m}-1$ do:
    - Set $I_i^{\mathsf{left},(1)} \leftarrow \mathbb{Z}_{\widetilde{m}}$, $I_i^{\mathsf{left},(2)} := I_i^{\mathsf{left},(1)}$;
    - Set $I_i^{\mathsf{left},(3)} = I_i^{\mathsf{left},(4)} := I_i^{\mathsf{left}} - I_i^{\mathsf{left},(1)} \pmod{\widetilde{m}}$;
    - Set $\mathcal{J}_i^{\mathsf{left},(1)} \leftarrow \mathbb{Z}_n$, $\mathcal{J}_i^{\mathsf{left},(2)} := \mathcal{J}_i^{\mathsf{left},(1)}$;
    - Set $\mathcal{J}_i^{\mathsf{left},(3)} = \mathcal{J}_i^{\mathsf{left},(4)} := \mathcal{J}_i^{\mathsf{left}} - \mathcal{J}_i^{\mathsf{left},(1)} \pmod{n}$;
    - Set $I_i^{\mathsf{right},(1)} \leftarrow \mathbb{Z}_{\widetilde{m}}$, $I_i^{\mathsf{right},(2)} := I_i^{\mathsf{right},(1)}$;
    - Set $I_i^{\mathsf{right},(3)} = I_i^{\mathsf{right},(4)} := I_i^{\mathsf{right}} - I_i^{\mathsf{right},(1)} \pmod{\widetilde{m}}$;
    - Set $\mathcal{J}_i^{\mathsf{right},(1)} \leftarrow \mathbb{Z}_n$, $\mathcal{J}_i^{\mathsf{right},(2)} := \mathcal{J}_i^{\mathsf{right},(1)}$;
    - Set $\mathcal{J}_i^{\mathsf{right},(3)} = \mathcal{J}_i^{\mathsf{right},(4)} := \mathcal{J}_i^{\mathsf{right}} - \mathcal{J}_i^{\mathsf{right},(1)} \pmod{n}$;
    - Set $t_i^{(1)} \leftarrow \mathbb{Z}_{2^\lambda}$, $t_i^{(2)} := t_i^{(1)}$;
    - Set $t_i^{(3)} = t_i^{(4)} := t_i - t_i^{(1)} \pmod{2^\lambda}$;
    - Set $v_i^{(1)} \leftarrow \mathbb{Z}_{2^\lambda}$, $v_i^{(2)} := v_i^{(1)}$;
    - Set $v_i^{(3)} = v_i^{(4)} := v_i - v_i^{(1)} \pmod{2^\lambda}$;
  - Set $\mathsf{id}_1 = 1$ and k_1 is the feature index of the source note;
  - Generate $\mathsf{id}_1^{(1)}, \ldots, \mathsf{id}_1^{(4)} \leftarrow \mathbb{Z}_{\widetilde{m}}$, $\mathsf{id}_1 = \sum_{i=1}^4 \mathsf{id}_1^{(i)} \pmod{m}$;
  - Generate $k_1^{(1)}, k_1^{(2)}, k_1^{(3)}, k_1^{(4)} \leftarrow \mathbb{Z}_n$, $k_1 = \sum_{i=1}^4 k_1^{(i)} \pmod{n}$;
  - Send $(\mathcal{P}^{(j)}, \mathsf{id}_1^{(j)}, k_1^{(j)})$ to S_j, $j \in [4]$;
- 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 $x_i^{(1)} \leftarrow \mathbb{Z}_{2^\ell}$, set $x_i^{(2)} := x_i^{(1)}$;
    - Set $x_i^{(3)} := x_i - x_i^{(1)} \pmod{2^\ell}$, $x_i^{(4)} := x_i^{(3)}$;
  - Send $\mathbf{x}^{(j)}$ to S_j, $j \in [4]$.

Figure 15: Outsourcing Protocol $\Pi_{\mathsf{os}}^{\mathsf{poly}}$.

v of all nodes in the evaluation path is exactly equal to the classification value of the reached leaf node.

Our polynomial-round protocol requires $2d$ rounds. Referring to the example in Fig. 14, 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:

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

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

Outsourcing. For polynomial-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. 15, for each $P_i \in \mathcal{P}$, the model owner M generates replicated shares $P_i^{(j)}$, $j \in [4]$. Send them to S_1, S_2 and S_3, S_4 respectively. 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 four servers in $(4,4)$-additive secret sharing.

Polynomial-round Evaluation Protocol $\Pi_{\mathsf{eval}}^{\mathsf{poly}}$

- Upon receiving $(\mathsf{Eval}, \mathsf{sid})$ from $\mathcal{Z}$, each server $S_j, j \in [4]$ **do**:
  - **for** $i := 1$ to d:
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(j)}, k_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathcal{P}^{(j)}, \mathsf{id}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{\widetilde{m},*}$ to get $P_{\mathsf{id}_i}^{(j)} := (I_{\mathsf{id}_i}^{\mathsf{left},(j)}, I_{\mathsf{id}_i}^{\mathsf{right},(j)}, \mathcal{J}_{\mathsf{id}_i}^{\mathsf{left},(j)}, \mathcal{J}_{\mathsf{id}_i}^{\mathsf{right},(j)}, t_{\mathsf{id}_i}^{(j)}, v_{\mathsf{id}_i}^{(j)})$[a];
    - Set $\mathsf{res}^{(j)} := \mathsf{res}^{(j)} + v_{\mathsf{id}_i}^{(j)} \pmod{2^\ell}$;
    - **if** $i \geq d$, return $\mathsf{res}^{(j)}$ to the receiver R and **break**;
    - Obliviously fetch $\mathsf{id}_{i+1}^{(j)}$ and $k_{i+1}^{(j)}$[b]:
      - Send $(\textsc{CompFetch}, \mathsf{sid}, (I_{\mathsf{id}_i}^{\mathsf{left},(j)}, I_{\mathsf{id}_i}^{\mathsf{right},(j)}), (x_{k_i}^{(j)}, t_{\mathsf{id}_i}^{(j)}))$ to $\mathcal{F}_{\mathsf{csot}}^{\log \widetilde{m},\ell}$
      - Send $(\textsc{CompFetch}, \mathsf{sid}, (\mathcal{J}_{\mathsf{id}_i}^{\mathsf{left},(j)}, \mathcal{J}_{\mathsf{id}_i}^{\mathsf{right},(j)}), (x_{k_i}^{(j)}, t_{\mathsf{id}_i}^{(j)}))$ 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 $P_{\mathsf{id}_i}^{(j)}$ by the same DPF keys but different output bit-lengths of evaluation. And all selections of i-th step are performed in the same round, including the feature selection of the previous row. Details of our oblivious selection subroutine can be found in Appendix E.

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

Figure 16: Polynomial-round Evaluation Protocol $\Pi_{\mathsf{eval}}^{\mathsf{poly}}$ in the $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$-hybrid model.

Evaluation. For i-th step in the evaluation, with the secret shared element index $\langle k_i \rangle$ and feature index $\langle \mathsf{id}_i \rangle$, servers invoke the 1-out-of-N shared OT protocol to fetch the feature $\langle x_{k_i} \rangle$ and the element $\langle P_{\mathsf{id}_i} \rangle$ in a parallel. For readability, we describe our protocol $\Pi_{\mathsf{eval}}^{\mathsf{poly}}$ in the $\{\mathcal{F}_{\mathsf{sot}}, \mathcal{F}_{\mathsf{csot}}\}$-hybrid model in Fig. 16, and the details of our real-world oblivious selection subroutine Π_{sel} can be found in Appendix E. Then servers 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 *res* to the requester, who is able to reconstruct the classification result locally.

Security. We show the security of our polynomial-round protocol ($\Pi_{\mathsf{os}}^{\mathsf{poly}}$, $\Pi_{\mathsf{eval}}^{\mathsf{poly}}$) with the following theorem, and its proof can be found in Appendix D.

Theorem 4. *The protocol $\Pi_{\mathsf{os}}^{\mathsf{poly}}$ as described in Fig. 15 and $\Pi_{\mathsf{eval}}^{\mathsf{poly}}$ as described in Fig. 16 UC-realizes $\mathcal{F}_{\mathsf{bp}}^4$ 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.*

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

7 Malicious Security

We can upgrade the proposed protocols to tolerate malicious adversary; that is the malicious behavior can be detected but without identifiable abort. First of all, as shown in [4], it is possible to modify the DPF and DCF schemes such that the correctness of the evaluation result can be tested when one of the evaluators are malicious. More specifically, we generate two pairs of FSS keys. One pair express the same point function (or less than function) f as in semi-honest protocol, and the other express $f_\alpha := \alpha f$, where $\alpha \leftarrow \mathbb{G}^{\mathsf{out}}$. To detect malicious FSS key generator behavior, we let the two non-evaluation servers (say S_1 and S_2) generate the same key pairs with a common random seed; therefore, the corresponding keys are identical. We let them independently send the FSS keys[1] to the evaluation servers S_3 and S_4, respectively. If one of the FSS key generators is malicious, the FSS keys would be different. We rest MPC operations can be upgraded to achieve malicious security using the linear MAC techniques presented in SPDZ protocols [9, 10].

8 Implementation and Benchmarks

The proposed constant-round scheme and polynomial-round scheme are implemented in C++. The DCF and DPF schemes are improved from [23]. Since Ma *et al.* [20] did not release their source code, we re-implement their scheme using AES-NI and EMP-toolkits [25]. In addition, the state-of-the-art constant-round protocols are adopted from [17] 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. There 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).

Our experiment uses datasets from the UCI machine learning repository [13], 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 [20], 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. Table 3 shows the offline phase performance comparison between our polynomial-round protocol and MTZC [20] in the outsourced setting. Compared with MTZC [20], our protocol is slightly slower for small DAG models, while it is about 4X faster for big DAG models.

Table 3: Offline phase running time comparison (ms) between our (2d-1)-round protocol and MTZC [20] in the outsourced setting. (Network setting: MAN (100Mbps/6ms RTT) and WAN (40Mbps/80ms RTT)

		Linnerud	Breast	Digits	Spambase
MAN	MTZC	4.319	5.190	102.4	177.5
	Ours(Poly)	9.14	13.21	41.54	46.32
WAN	MTZC	7.541	11.56	615.9	879.9
	Ours(Poly)	9.88	15.26	164.7	196.9

Fig.17 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 polynomial-round protocol runs much more faster than the state-of-the-arts. More precisely, our polynomial-round protocol is up to 15X faster than the others in the LAN setting.

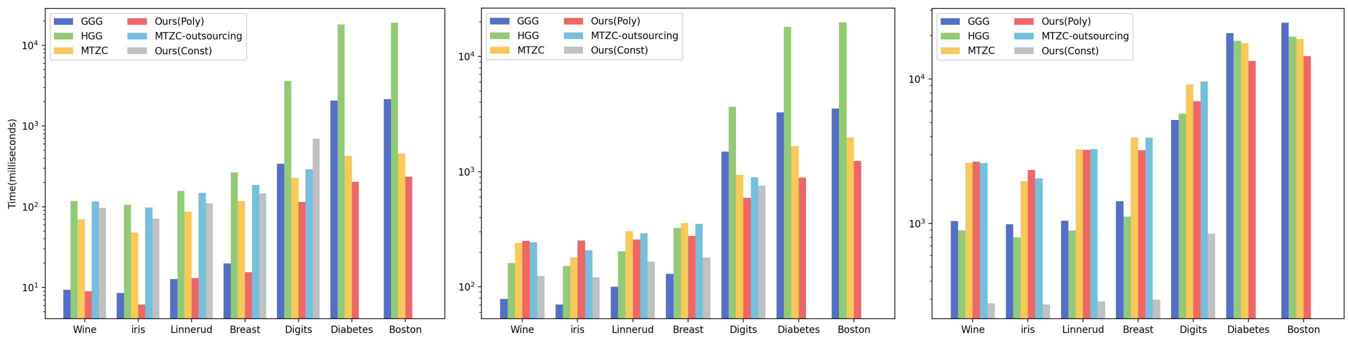
In a network environment with lower bandwidth and higher latency such as the WAN setting, our constant-round protocol outperforms the state-of-the-art protocols. In particular, our constant-round protocol is up to 10X faster than the others in the WAN setting.

9 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 [15]. 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.* [7] present a private diagnosis system based on BP model. They imple-

[1] One of them could be hash for efficiency.



(a) LAN(1Gbps/0.1ms) (b) MAN(100Mbps/6ms) (c) WAN(40Mbps/80ms)

Figure 17: Online runtime (in different log scales) in LAN/MAN/WAN (bandwidth/RTT) setting with Intel(R) Core i7 8700 CPU @ 3.2 GHz running Ubuntu 18.04.2 LTS, 32 GB Memory and 1TB SSD. Ours(Poly)/Ours(Const) refers to our polynomial-round/constant-round protocol. MTZC [20] refers to the sparse tree variant; MTZC-Outsourcing refers to their outsourcing variant.

ment 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. [3] 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. [26] gets rid of FHE by using DGK protocol based on AHE for comparison and OT for leaf node selection. But [26] requires a complete tree (with dummy nodes) and permuting it. [21] improves [26] 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, [26] and [21] take advantage of the properties of the tree structure, thus no longer support BP evaluation. [18] 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. [18] 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 [16] and [22]. 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[2]. The protocol of [11] 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. [27] is inspired by [11], and use additive secret sharing. [27] introduces a standard modulus conversion after bit-wise comparison and follows the path cost computation of [21]. The protocol of [27] has the same communication complexity as [11]. The state-of-the-art work of outsourced evaluation protocol is from [20]. 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 [20] 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.

10 Concluding Remarks

We presented a 4-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 [12], 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 polynomial-round outsourced private decision tree evaluation protocol achieves

[2]We consider the secure outsourcing without the leakage of the index mapping between decision nodes and input features. [19] and [1] do not meet this condition.

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. As a future work, we will extend our 4-server MPC platform to support RAM programs.

References

[1] Asma Aloufi, Peizhao Hu, Harry W. H. Wong, and Sherman S. M. Chow. Blindfolded evaluation of random forests with multi-key homomorphic encryption. *IEEE Trans. Dependable Secur. Comput.*, 18(4):1821–1835, 2019.

[2] Mauro Barni, Pierluigi Failla, Vladimir Kolesnikov, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. Secure evaluation of private linear branching programs with medical applications. In *ESORICS 2009*, volume 5789 of *LNCS*, pages 424–439, Saint-Malo, France, September 21–23, 2009. Springer.

[3] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *NDSS 2015*, San Diego, CA, USA, February 8–11, 2015. The Internet Society.

[4] Elette Boyle, Nishanth Chandran, Niv Gilboa, Divya Gupta, Yuval Ishai, Nishant Kumar, and Mayank Rathee. Function secret sharing for mixed-mode and fixed-point secure computation. In *EUROCRYPT 2021, Part II*, volume 12697 of *LNCS*, pages 871–900, Zagreb, Croatia, October 17–21, 2021. Springer.

[5] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 337–367, Sofia, Bulgaria, April 26–30, 2015. Springer.

[6] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In *ACM CCS 2016*, pages 1292–1303, Vienna, Austria, October 24–28, 2016. ACM Press.

[7] Justin Brickell, Donald E. Porter, Vitaly Shmatikov, and Emmett Witchel. Privacy-preserving remote diagnostics. In *ACM CCS 2007*, pages 498–507, Alexandria, Virginia, USA, October 28–31, 2007. ACM Press.

[8] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS'01*, pages 136–145. IEEE, 2001.

[9] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In *ESORICS*, pages 1–18, 2013.

[10] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *CRYPTO*, pages 643–662, 2012.

[11] Martine De Cock, Rafael Dowsley, Caleb Horst, Raj Katti, Anderson C. A. Nascimento, Wing-Sea Poon, and Stacey Truex. Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation. *IEEE Trans. Dependable Secur. Comput.*, 16(2):217–230, February 2017.

[12] Jack Doerner and abhi shelat. Scaling ORAM for secure computation. In *ACM CCS 2017*, pages 523–535, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

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

[14] S. Dov Gordon, Jonathan Katz, Vladimir Kolesnikov, Fernando Krell, Tal Malkin, Mariana Raykova, and Yevgeniy Vahlis. Secure two-party computation in sublinear (amortized) time. In *ACM CCS 2012*, pages 513–524, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

[15] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *TCC 2007*, volume 4392 of *LNCS*, pages 575–594, Amsterdam, The Netherlands, February 21–24, 2007. Springer.

[16] Marc Joye and Fariborz Salehi. Private yet efficient decision tree evaluation. In *DBSec 2018*, volume 10980, pages 243–259, Bergamo, Italy, July 16–18 2018. Springer.

[17] Ágnes Kiss, Masoud Naderpour, Jian Liu, N. Asokan, and Thomas Schneider. Private Decision Tree Evaluation (PDTE) Protocols. https://github.com/encryptogroup/PDTE, 2019.

[18] Ágnes Kiss, Masoud Naderpour, Jian Liu, N. Asokan, and Thomas Schneider. SoK: Modular and efficient private decision tree evaluation. *PoPETs*, 2019(2):187–208, April 2019.

[19] Lin Liu, Jinshu Su, Rongmao Chen, Jinrong Chen, Guangliang Sun, and Jie Li. Secure and fast decision tree evaluation on outsourced cloud data. In *ML4CS 2019*, pages 361–377, Xi'an, China, September 19–21, 2019. Springer.

[20] Jack P. K. Ma, Raymond K. H. Tai, Yongjun Zhao, and Sherman S.M. Chow. Let's stride blindfolded in a forest: Sublinear multi-client decision trees evaluation. In *NDSS 2021*, San Diego, CA, USA, February 21-25, 2021. The Internet Society.

[21] Raymond K. H. Tai, Jack P. K. Ma, Yongjun Zhao, and Sherman S. M. Chow. Privacy-preserving decision trees evaluation via linear functions. In *ESORICS 2017, Part II*, volume 10493 of *LNCS*, pages 494–512, Oslo, Norway, September 11–15, 2017. Springer.

[22] Anselme Tueno, Florian Kerschbaum, and Stefan Katzenbeisser. Private evaluation of decision trees using sublinear cost. *PoPETs*, 2019(1):266–286, January 2019.

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

[24] Xiao Wang, T.-H. Hubert Chan, and Elaine Shi. Circuit ORAM: On tightness of the Goldreich-Ostrovsky lower bound. In *ACM CCS 2015*, pages 850–861, Denver, CO, USA, October 12–16, 2015. ACM Press.

[25] Xiao Wang, Alex J. Malozemoff, and Jonathan Katz. EMP-toolkit: Efficient MultiParty computation toolkit. https://github.com/emp-toolkit, 2016.

[26] David J. Wu, Tony Feng, Michael Naehrig, and Kristin E. Lauter. Privately evaluating decision trees and random forests. *PoPETs*, 2016(4):335–355, October 2016.

[27] Yifeng Zheng, Huayi Duan, and Cong Wang. Towards secure and efficient outsourcing of machine learning classification. In *ESORICS 2019, Part I*, volume 11735 of *LNCS*, pages 22–40, Luxembourg, September 23–27, 2019. Springer.

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. 5 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

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

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 $\mathcal{S} := \{S_1,\ldots,S_4\}$ 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 $S_1,\ldots,S_4$ interact with functionality $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$ in the ideal world, and corrupted parties are controlled by the simulator Sim. We consider following cases.

Case 1: S_1 (or 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 honest parties S_2,S_3,S_4. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon initialization, the simulator Sim acts as the honest party S_3 to do:
  - Generate $\varphi_1,\varphi_1^{(2)} \leftarrow \mathbb{Z}_N$ and set $\varphi_1^{(1)} := \varphi_1 - \varphi_1^{(2)}$;
  - Set $\mathcal{K}_{\varphi_1}^{(1)},\mathcal{K}_{\varphi_1}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda,\varphi_1,1,\mathbb{Z}_N,\mathbb{Z}_{2^\ell})$;
  - Send $\mathcal{K}_{\varphi_1}^{(1)}$ to S_1, $\mathcal{K}_{\varphi_1}^{(2)}$ to S_2;
- The simulator Sim does:
  - Pick random $w_1^{(1)},w_1^{(2)} \leftarrow \mathbb{Z}_N$;
  - Set $w_1^{(3)} := w_1^{(1)}$, $w_1^{(4)} := w_1^{(2)}$;
- 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:
  - Set $\delta_1^{(j)} := w_1^{(j)}$ and $\delta_2^{(j)} := 0$;
  - Send $\delta_1^{(j)}$ to S_{3-j}, $\delta_2^{(j)}$ to S_3 on behave of S_j;
- Upon receiving $(\textsc{Fetch},\mathsf{sid},S_j)$ for an honest party S_j, $j \in \{3,4\}$ from the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$, the simulator Sim does:
  - Set $\delta_1^{(j)} := -\varphi_1^{(j-2)} - w_1^{(j)}$ and $\delta_2^{(j)} := 0$;
  - Send $\delta_1^{(j)}$ to S_1 and S_2, $\delta_2^{(j)}$ to S_{7-j} on behave of S_j;
- Upon receiving $\delta_1^{(1)}$ from the corrupted S_1 to S_2 and $\delta_2^{(1)}$ from the corrupted S_1 to S_3,S_4, the simulator Sim does:
  - Extract $i^{(1)} := \delta_1^{(1)} - \mathsf{PRF}_{\eta_1}(\mathsf{sid},1) \pmod N$;
  - Send $(\textsc{Fetch},\mathsf{sid},\mathbf{x}^{(2)},i^{(1)})$ to the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$;
  - Compute $\delta_1 := \delta_1^{(1)} + \delta_1^{(2)} + \delta_1^{(3)} + \delta_1^{(4)} \pmod N$;

– Set $\tilde{x}^{(1)}_k := x^{(2)}_{k-\delta_1 \pmod N}$, for $k \in \mathbb{Z}_N$;
– Set $(\beta^{(1)}_{k,\varphi_1})_{k\in\mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(1, \mathcal{K}^{(1)}_{\varphi_1})$;
– Set $\zeta_1 \leftarrow \mathsf{PRF}^{\mathbb{Z}_{2^\ell}}_{\eta_1}(\mathsf{sid})$, $\zeta_3 \leftarrow \mathsf{PRF}^{\mathbb{Z}_{2^\ell}}_{\eta_3}(\mathsf{sid})$;
– Compute $y^{(1)} := \sum_{k=0}^{N-1}(\tilde{x}^{(1)}_k \cdot \beta^{(1)}_{k,\varphi_1}) + \zeta_1 - \zeta_3 \pmod{2^\ell}$.
– Send $(\textsc{Rand}, \mathsf{sid}, y^{(1)})$ to the external $\mathcal{F}^{N,\ell}_{\mathsf{sot}}$;

Indistinguishability. We assume that the parties $S_1,\ldots,S_4$ 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 $\mathsf{Exec}_{\Pi^{N,\ell}_{\mathsf{sot}},\mathcal{A},\mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w^{(j)}_1\}_{j\in[2]}$ and $\varphi^{(2)}_1$ are picked uniformly random from $\mathbb{Z}_N$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_N}$. Set $w^{(3)}_1 := w^{(1)}_1$, $w^{(4)}_1 := w^{(2)}_1$.

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 $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\varepsilon_1 := 3\cdot\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda,\mathcal{A})$.*

Proof. We have changed 3 PRF outputs to uniformly random strings; therefore, the overall advantage is $3\cdot\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda,\mathcal{A})$ 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 $j\in\{1,2\}$, set $\tilde{\delta}^{(j)}_1 := w^{(j)}_1$ and $\tilde{\delta}^{(j)}_2 := 0$.
- For $j\in\{3,4\}$, set $\tilde{\delta}^{(j)}_1 := -\varphi^{(j-2)}_1 - w^{(j)}_1$ and $\tilde{\delta}^{(j)}_2 := 0$.

instead of
- For $j\in\{1,2\}$:
  – Set $\delta^{(j)}_1 := i^{(j)} + w^{(j)}_1$ and $\delta^{(j)}_2 := i^{(j)} - \varphi^{(j)}_2 + w^{(j)}_2$.
- For $j\in\{3,4\}$:
  – set $\delta^{(j)}_1 := i^{(j)} - \varphi^{(j-2)}_1 - w^{(j)}_1$ and $\delta^{(j)}_2 := i^{(j)} - w^{(j)}_2$.

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 $\mathsf{Adv}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}_{\mathsf{DPF}}(1^\lambda,\mathcal{A})$, then $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\varepsilon_2 := \mathsf{Adv}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}_{\mathsf{DPF}}(1^\lambda,\mathcal{A})$.*

Proof. Note that the corrupted party S_1 only sees $\{\delta^{(j)}_1\}_{j\in[4]}$; therefore, the modification of $\{\delta^{(j)}_2\}_{j\in[4]}$ is oblivious to S_1. In the hybrid $\mathcal{H}_1$, we have
- $\delta^{(1)}_1 := i^{(1)} + w^{(1)}_1$;
- $\delta^{(2)}_1 := i^{(2)} + w^{(2)}_1$;
- $\delta^{(3)}_1 := i^{(1)} - \varphi^{(1)}_1 - w^{(3)}_1$;
- $\delta^{(4)}_1 := i^{(1)} - \varphi^{(2)}_1 - w^{(4)}_1$;

It is straightforward that the distribution of $\{\delta^{(j)}_1\}_{j\in[4]}$ are uniformly random under the condition $\delta_1 := \sum_{k=1}^4 \delta^{(k)}_1 = i - \varphi_1$, where φ_1 is used to generate the DPF keys $\mathcal{K}^{(1)}_{\varphi_1}, \mathcal{K}^{(2)}_{\varphi_1} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \varphi_1, 1, \mathbb{Z}_N, \mathbb{Z}_{2^\ell})$. Whereas $\delta_1 := -\varphi_1$ 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 $\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}} := (\mathsf{Gen},\mathsf{Eval})$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\varepsilon_2 := \mathsf{Adv}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}_{\mathsf{DPF}}(1^\lambda,\mathcal{A})$. ☐

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

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

Case 2: S_3 (or S_4) is corrupted.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of honest parties S_1, S_2, S_4. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.
- Upon initialization, the simulator Sim acts as the honest party S_1 to do:
  – Generate $\varphi_2, \varphi^{(2)}_2 \leftarrow \mathbb{Z}_N$ and set $\varphi^{(1)}_2 := \varphi_2 - \varphi^{(2)}_2$;
  – Set $\mathcal{K}^{(1)}_{\varphi_2}, \mathcal{K}^{(2)}_{\varphi_2} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \varphi_2, 1, \mathbb{Z}_N, \mathbb{Z}_{2^\ell})$;
  – Send $\mathcal{K}^{(1)}_{\varphi_2}$ to S_3, $\mathcal{K}^{(2)}_{\varphi_2}$ to S_4;
- The simulator Sim does:
  – Pick random $w^{(1)}_2, w^{(2)}_2 \leftarrow \mathbb{Z}_N$;
  – Set $w^{(3)}_2 := w^{(1)}_2$, $w^{(4)}_2 := w^{(2)}_2$;
- Upon receiving $(\textsc{Fetch}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1,2\}$ from the external $\mathcal{F}^{N,\ell}_{\mathsf{sot}}$, the simulator Sim does:
  – Set $\delta^{(j)}_1 := 0$ and $\delta^{(j)}_2 := w^{(j)}_2 - \varphi^{(j)}_2$;
  – Send $\delta^{(j)}_1$ to S_{3-j}, $\delta^{(j)}_2$ to S_3 on behave of S_j;
- Upon receiving $(\textsc{Fetch}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{3,4\}$ from the external $\mathcal{F}^{N,\ell}_{\mathsf{sot}}$, the simulator Sim does:
  – Set $\delta^{(j)}_1 := 0$ and $\delta^{(j)}_2 := -w^{(j)}_2$;
  – Send $\delta^{(j)}_1$ to S_1 and S_2, $\delta^{(j)}_2$ to S_{7-j} on behave of S_j;
- Upon receiving $\delta^{(3)}_1$ from the corrupted S_3 to S_1, S_2 and $\delta^{(3)}_2$ from the corrupted S_3 to S_4, the simulator Sim does:
  – Extract $i^{(3)} := \delta^{(3)}_2 + \mathsf{PRF}_{\eta_1}(\mathsf{sid}, 1) \pmod N$;
  – Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^{(4)}, i^{(3)})$ to the external $\mathcal{F}^{N,\ell}_{\mathsf{sot}}$;
  – Compute $\delta_2 := \delta^{(1)}_2 + \delta^{(2)}_2 + \delta^{(3)}_2 + \delta^{(4)}_2 \pmod N$;
  – Set $\tilde{x}^{(3)}_k := x^{(4)}_{k-\delta_2 \pmod N}$, for $k \in \mathbb{Z}_N$;
  – Set $(\beta^{(1)}_{k,\varphi_2})_{k\in\mathbb{Z}_N} \leftarrow \mathsf{DPF.EvalAll}(1, \mathcal{K}^{(1)}_{\varphi_2})$;
  – Set $\zeta_1 \leftarrow \mathsf{PRF}^{\mathbb{Z}_{2^\ell}}_{\eta_1}(\mathsf{sid})$, $\zeta_4 \leftarrow \mathsf{PRF}^{\mathbb{Z}_{2^\ell}}_{\eta_4}(\mathsf{sid})$;
  – Compute $y^{(3)} := \sum_{k=0}^{N-1}(\tilde{x}^{(1)}_k \cdot \beta^{(1)}_{k,\varphi_1}) - \zeta_1 - \zeta_4 \pmod{2^\ell}$.
  – Send $(\textsc{Rand}, \mathsf{sid}, y^{(3)})$ to the external $\mathcal{F}^{N,\ell}_{\mathsf{sot}}$;

Indistinguishability. We assume that the parties $S_1,\ldots,S_4$ 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 $\mathsf{Exec}_{\Pi^{N,\ell}_{\mathsf{sot}},\mathcal{A},\mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w_2^{(j)}\}_{j\in[2]}$ and $\varphi_2^{(2)}$ are picked uniformly random from $\mathbb{Z}_N$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_N}$. Set $w_2^{(3)} := w_2^{(1)}$, $w_2^{(4)} := w_2^{(2)}$.

Claim 3. *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 $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda,\mathcal{A})$, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\varepsilon_1 := 3\cdot\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda,\mathcal{A})$.*

Proof. We have changed 3 PRF outputs to uniformly random strings; therefore, the overall advantage is $3\cdot\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_N}}(1^\lambda,\mathcal{A})$ 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 $j\in\{1,2\}$, set $\tilde{\delta}_1^{(j)} := 0$ and $\tilde{\delta}_2^{(j)} := w_2^{(j)} - \varphi_2^{(j)}$.
- For $j\in\{3,4\}$, set $\tilde{\delta}_1^{(j)} := 0$ and $\tilde{\delta}_2^{(j)} := -w_2^{(j)}$.

instead of
- For $j\in\{1,2\}$:
  - Set $\delta_1^{(j)} := i^{(j)} + w_1^{(j)}$ and $\delta_2^{(j)} := i^{(j)} - \varphi_2^{(j)} + w_2^{(j)}$.
- For $j\in\{3,4\}$:
  - set $\delta_1^{(j)} := i^{(j)} - \varphi_1^{(j-2)} - w_1^{(j)}$ and $\delta_2^{(j)} := i^{(j)} - w_2^{(j)}$.

Claim 4. *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 $\varepsilon_2 := \mathsf{Adv}_{\mathsf{DPF}}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}(1^\lambda,\mathcal{A})$.*

Proof. Note that the corrupted party S_3 only sees $\{\delta_2^{(j)}\}_{j\in[4]}$; therefore, the modification of $\{\delta_1^{(j)}\}_{j\in[4]}$ is oblivious to S_3. In the hybrid $\mathcal{H}_1$, we have
- $\delta_2^{(1)} := i^{(1)} - \varphi_2^{(1)} + w_2^{(1)}$;
- $\delta_2^{(2)} := i^{(2)} - \varphi_2^{(2)} + w_2^{(2)}$;
- $\delta_2^{(3)} := i^{(1)} - w_2^{(3)}$;
- $\delta_2^{(4)} := i^{(1)} - w_2^{(4)}$;

It is straightforward that the distribution of $\{\delta_1^{(j)}\}_{j\in[4]}$ are uniformly random under the condition $\delta_2 := \sum_{k=1}^4 \delta_2^{(k)} = i - \varphi_2$, where φ_2 is used to generate the DPF keys $\mathcal{K}_{\varphi_2}^{(1)},\mathcal{K}_{\varphi_2}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda,\varphi_2,1,\mathbb{Z}_N,\mathbb{Z}_{2^\ell})$. Whereas $\delta_2 := -\varphi_2$ 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 $\mathsf{DPF}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}} := (\mathsf{Gen},\mathsf{Eval})$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\varepsilon_2 := \mathsf{Adv}_{\mathsf{DPF}}^{\mathbb{Z}_N,\mathbb{Z}_{2^\ell}}(1^\lambda,\mathcal{A})$. □

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

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

This concludes the proof. □

B Proof of Theorem 2

Theorem 2. Let $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^{\ell_1}},\mathbb{Z}_{2^{\ell_2}}}$ be a secure function secret sharing scheme for offset comparison function $f^{\mathsf{IC}}_{\alpha,\beta}(x) : \mathbb{Z}_{2^{\ell_2}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda,\mathcal{A})$. 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 $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda,\mathcal{A})$. Let $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}} : \{0,1\}^\lambda\times\{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_2}}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda,\mathcal{A})$. The protocol $\Pi^{\ell_1,\ell_2}_{\mathsf{csot}}$ as described in Fig. 7 UC-realizes $\mathcal{F}^{\ell_1,\ell_2}_{\mathsf{csot}}$ as described in Fig. 6 against semi-honest adversaries who can statically corrupted up to 1 server with distinguishing advantage

$$8\cdot\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda,\mathcal{A}) + 3\cdot\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda,\mathcal{A}) + \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda,\mathcal{A}).$$

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 $\mathsf{Exec}_{\Pi^{\ell_1,\ell_2}_{\mathsf{csot}},\mathcal{A},\mathcal{Z}}$ where the parties $\mathcal{S} := \{S_1,\ldots,S_4\}$ run protocol $\Pi^{\ell_1,\ell_2}_{\mathsf{csot}}$ 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 $\mathsf{Exec}_{\mathcal{F}^{\ell_1,\ell_2}_{\mathsf{csot}},\mathsf{Sim},\mathcal{Z}}$ where the parties $S_1,\ldots,S_4$ interact with functionality $\mathcal{F}^{\ell_1,\ell_2}_{\mathsf{csot}}$ in the ideal world, and corrupted parties are controlled by the simulator Sim. We consider following cases.

Case 1: S_1 (or 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 honest parties S_2,S_3,S_4. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.
- Upon initialization, the simulator Sim acts as the honest party S_3 to do:
  - Generate $\rho_1,\rho_1^{(2)} \leftarrow \mathbb{Z}_{2^{\ell_2}}$ and set $\rho_1^{(1)} := \rho_1 - \rho_1^{(2)}$;
  - $\mathcal{K}_{\rho_1}^{(1)},\mathcal{K}_{\rho_1}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda,2^{\ell_2-1},1,\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}},\rho_1)$;
  - Send $\mathcal{K}_{\rho_1}^{(1)}$ to S_1, $\mathcal{K}_{\rho_1}^{(2)}$ to S_2;
- The simulator Sim does:
  - Pick random $w_{x,0}^{(1)},w_{x,0}^{(2)},w_{x,1}^{(1)},w_{x,1}^{(2)},r_1,r_2,r_3,r_4 \leftarrow \mathbb{Z}_{2^{\ell_1}}$;

– Pick random $w_{m,1}^{(1)}, w_{m,1}^{(2)}, \leftarrow \mathbb{Z}_{2^{\ell_2}}$;
– Set $w_{x,0}^{(3)} := w_{x,0}^{(1)}$, $w_{x,0}^{(4)} := w_{x,0}^{(2)}$, $w_{x,1}^{(3)} := w_{x,1}^{(1)}$, $w_{x,1}^{(4)} := w_{x,1}^{(2)}$, $w_{m,1}^{(3)} := w_{m,1}^{(1)}$, $w_{m,1}^{(4)} := w_{m,1}^{(2)}$;

- Upon receiving $(\textsc{CompFetch}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1,2\}$ from the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$, Sim does:
  – Set $\Delta m_{\rho_1}^{(j)} := w_{m,1}^{(j)}$, $\Delta m_{\rho_2}^{(j)} := 0$;
  – Set $\tilde{x}_i^{(j)} := w_{x,i}^{(j)}$, for $i \in \{0,1\}$;
  – Send $(\Delta m_{\rho_1}^{(j)}, \tilde{\mathbf{x}}^{(j)})$ to S_{3-j}, $\Delta m_{\rho_2}^{(j)}$ to S_3 and S_4 on behalf of the honest party S_j;
- Upon receiving $(\textsc{CompFetch}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{3,4\}$ from the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$, Sim does:
  – Set $\hat{x}_0^{(j)} := r_1 + r_2 \pmod{2^{\ell_1}}$, $\hat{x}_1^{(j)} := r_3 + r_4 \pmod{2^{\ell_1}}$;
  – Set $\Delta m_{\rho_1}^{(j)} := \rho_1^{(j-2)} - w_{m,1}^{(j)} \pmod{2^{\ell_2}}$, $\Delta m_{\rho_2}^{(j)} := 0$;
  – Set $\tilde{x}_i^{(j)} := -r_{j+2i-2} - w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, $i \in \{0,1\}$;
  – Send $\{\Delta m_{\rho_1}^{(j)}, \tilde{\mathbf{x}}^{(j)}\}$ to S_1 and S_2, $\Delta m_{\rho_2}^{(j)}$ to S_{7-j};
- Upon receiving $(\Delta m_{\rho_1}^{(1)}, \tilde{\mathbf{x}}^{(1)})$ from the corrupted S_1 to S_2 and $\Delta m_{\rho_2}^{(1)}$ from the corrupted S_1 to S_3, S_4, the simulator Sim does:
  – For $i \in \{0,1\}$, extract $x_i^{(1)} := \tilde{x}_i^{(1)} - w_{x,i}^{(3)} \pmod{2^{\ell_1}}$;
  – Extract $\Delta m^{(1)} := \Delta m_{\rho_1}^{(1)} + w_{m,1}^{(3)} \pmod{2^{\ell_2}}$;
  – Pick random $m_0^{(1)}, m_1^{(1)} \leftarrow \mathbb{Z}_{2^{\ell_2}}$ s.t. $m_1^{(1)} - m_0^{(1)} = \Delta m^{(1)}$;
  – Send $(\textsc{CompFetch}, \mathsf{sid}, \mathbf{x}^{(1)}, \mathbf{m}^{(1)})$ to the external $\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2}$;
  – Set $\Delta m_{\rho_1} := \sum_{i=1}^{4} \Delta m_{\rho_1}^{(i)} \pmod{2^{\ell_2}}$;
  – Set $\hat{x}_0^{(1)} := \sum_{i=1}^{4} \tilde{x}_0^{(i)} \pmod{2^{\ell_1}}$;
  – Set $\hat{x}_1^{(1)} := \sum_{i=1}^{4} \tilde{x}_1^{(i)} \pmod{2^{\ell_1}}$;
  – Set $\beta_1^{(1)} \leftarrow \mathsf{DCF.Eval}^{\mathsf{IC}}(1, \mathcal{K}_{\rho_1}^{(1)}, \Delta m_{\rho_1})$;
  – Set $\zeta_1 \leftarrow \mathsf{PRF}_{\eta_1}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid}, 3)$, $\zeta_3 \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid}, 1)$;
  – Compute $y^{(1)} := \beta_1^{(1)} \cdot (\hat{x}_0^{(1)} - \hat{x}_1^{(1)}) + \zeta_1 - \zeta_3$;
  – Send $(\textsc{Rand}, \mathsf{sid}, y^{(1)})$ to the external $\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2}$;

Indistinguishability. We assume that the parties $S_1, \ldots, S_4$ 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 $\mathsf{Exec}_{\Pi_{\mathsf{csot}}^{\ell_1,\ell_2}, \mathcal{A}, \mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w_{x,0}^{(j)}\}_{j\in[2]}$, $\{w_{x,1}^{(j)}\}_{j\in[2]}$ and $\{r_j\}_{j\in[4]}$ are picked uniformly random from $\mathbb{Z}_{2^{\ell_1}}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}$; $\{w_{m,1}^{(j)}\}_{j\in[2]}$ and $\rho_1^{(2)}$ are picked uniformly random from $\mathbb{Z}_{2^{\ell_2}}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}$. Set $w_{x,0}^{(3)} := w_{x,0}^{(1)}$, $w_{x,0}^{(4)} := w_{x,0}^{(2)}$, $w_{x,1}^{(3)} := w_{x,1}^{(1)}$, $w_{x,1}^{(4)} := w_{x,1}^{(2)}$, $w_{m,1}^{(3)} := w_{m,1}^{(1)}$, $w_{m,1}^{(4)} := w_{m,1}^{(2)}$.

Claim 5. *If $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$, $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_2}}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_1$ and $\mathcal{H}_0$ are indistinguishable with advantage $\varepsilon_1 := 8 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A})$.*

Proof. We have changed 8 $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}$ outputs and 3 $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}$ outputs to uniformly random strings; therefore, the overall advantage is $8 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A})$ 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 $j \in \{1,2\}$:
  – Set $\Delta m_{\rho_1}^{(j)} := w_{m,1}^{(j)}$ and $\Delta m_{\rho_2}^{(j)} := 0$;
- For $j \in \{3,4\}$:
  – Set $\Delta m_{\rho_1}^{(j)} := \rho_1^{(j-2)} - w_{m,1}^{(j)} \pmod{2^{\ell_2}}$, $\Delta m_{\rho_2}^{(j)} := 0$;

instead of

- For $j \in \{1,2\}$:
  – Set $\Delta m_{\rho_1}^{(j)} := m_1^{(j)} - m_0^{(j)} + w_{m,1}^{(j)} \pmod{2^{\ell_2}}$;
  – Set $\Delta m_{\rho_2}^{(j)} := m_1^{(j)} - m_0^{(j)} + \rho_2^{(j)} + w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;
- For $j \in \{3,4\}$:
  – Set $\Delta m_{\rho_1}^{(j)} := m_1^{(j)} - m_0^{(j)} + \rho_1^{(j-2)} - w_{m,1}^{(j)} \pmod{2^{\ell_2}}$;
  – Set $\Delta m_{\rho_2}^{(j)} := m_1^{(j)} - m_0^{(j)} - w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;

Claim 6. *If $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}} := (\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ is a secure function secret sharing scheme for offset comparison function $f_{\alpha,\beta}^{\mathsf{IC}}(x) : \mathbb{Z}_{2^{\ell_2}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage $\varepsilon_2 := \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A})$.*

Proof. Note that the corrupted party S_1 only sees $\{\Delta m_{\rho_1}^{(j)}\}_{j\in[4]}$; therefore, the modification of $\{\Delta m_{\rho_2}^{(j)}\}_{j\in[4]}$ is oblivious to S_1. In the hybrid $\mathcal{H}_1$, we have

- $\Delta m_{\rho_1}^{(1)} := m_1^{(1)} - m_0^{(1)} + w_{m,1}^{(1)} \pmod{2^{\ell_2}}$;
- $\Delta m_{\rho_1}^{(2)} := m_1^{(2)} - m_0^{(2)} + w_{m,1}^{(2)} \pmod{2^{\ell_2}}$;
- $\Delta m_{\rho_1}^{(3)} := m_1^{(3)} - m_0^{(3)} + \rho_1^{(1)} - w_{m,1}^{(3)} \pmod{2^{\ell_2}}$;
- $\Delta m_{\rho_1}^{(4)} := m_1^{(4)} - m_0^{(4)} + \rho_1^{(2)} - w_{m,1}^{(4)} \pmod{2^{\ell_2}}$;

It is straightforward that the distribution of $\{\Delta m_{\rho_1}^{(j)}\}_{j\in[4]}$ are uniformly random under the condition $\Delta m_{\rho_1} := \sum_{k=1}^{4} \Delta m_{\rho_1}^{(k)} = m_1 - m_0 + \rho_1$, where ρ_1 is used to generate the DCF keys $\mathcal{K}_{\rho_1}^{(1)}, \mathcal{K}_{\rho_1}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell_2 - 1}, 1, \mathbb{Z}_{2^{\ell_2}}, \mathbb{Z}_{2^{\ell_1}}, \rho_1)$. Whereas $\Delta m_1 := \rho_1$ 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 $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}} := (\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\varepsilon_2 := \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A})$. □

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

- For $j \in \{1,2\}$:
  - Set $\tilde{x}_i^{(j)} := w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, $i \in \{0,1\}$;
- For $j \in \{3,4\}$:
  - Set $\tilde{x}_i^{(j)} := -r_{j+2i-2} - w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, $i \in \{0,1\}$;

instead of

- For $j \in \{1,2\}$:
  - Set $\tilde{x}_i^{(j)} := x_i^{(j)} + w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, for $i \in \{0,1\}$;
- For $j \in \{3,4\}$:
  - Set $\tilde{x}_i^{(j)} := x_i^{(j)} - r_{j+2i-2} - w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, $i \in \{0,1\}$;

Claim 7. *$\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable.*

Proof. Since $\{r_i\}_{i\in[4]}$ and $\{w_{x,0}^{(j)}, w_{x,1}^{(j)}\}_{j\in[2]}$ are uniformly random in $\mathbb{Z}_{2^{\ell_1}}$, the distribution of $\{\tilde{x}_0^{(j)}, \tilde{x}_1^{(j)}\}_{j\in[4]}$ and $\{x_0^{(j)}, x_1^{(j)}\}_{j\in[4]}$ 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 $\mathsf{Exec}_{\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2},\mathcal{S},\mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$8 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A}) + \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A}).$$

Case 2: S_3 (or S_4) is corrupted.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of honest parties S_1, S_2, S_4. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon initialization, the simulator Sim acts as the honest party S_1 to do:
  - Generate $\rho_2, \rho_2^{(2)} \leftarrow \mathbb{Z}_{2^{\ell_2}}$ and set $\rho_2^{(1)} := \rho_2 - \rho_2^{(2)}$;
  - $\mathcal{K}_{\rho_2}^{(1)}, \mathcal{K}_{\rho_2}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell_2-1}, 1, \mathbb{Z}_{2^{\ell_2}}, \mathbb{Z}_{2^{\ell_1}}, \rho_2)$;
  - Send $\mathcal{K}_{\rho_2}^{(1)}$ to S_3, $\mathcal{K}_{\rho_2}^{(2)}$ to S_4;
- The simulator Sim does:
  - Pick random $w_{x,0}^{(1)}, w_{x,0}^{(2)}, w_{x,1}^{(1)}, w_{x,1}^{(2)}, r_1, r_2, r_3, r_4 \leftarrow \mathbb{Z}_{2^{\ell_1}}$;
  - Pick random $w_{m,2}^{(1)}, w_{m,2}^{(2)}, \leftarrow \mathbb{Z}_{2^{\ell_2}}$;
  - Set $w_{x,0}^{(3)} := w_{x,0}^{(1)}$, $w_{x,0}^{(4)} := w_{x,0}^{(2)}$, $w_{x,1}^{(3)} := w_{x,1}^{(1)}$, $w_{x,1}^{(4)} := w_{x,1}^{(2)}$, $w_{m,2}^{(3)} := w_{m,2}^{(1)}$, $w_{m,2}^{(4)} := w_{m,2}^{(2)}$;
- Upon receiving $(\textsc{CompFetch}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1,2\}$ from the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$, Sim does:
  - Set $\Delta m_{\rho_1}^{(j)} := 0$, $\Delta m_{\rho_2}^{(j)} := \rho_2^{(j)} + w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\tilde{x}_i^{(j)} := w_{x,i}^{(j)}$, for $i \in \{0,1\}$;
  - Send $(\Delta m_{\rho_1}^{(j)}, \tilde{\mathbf{x}}^{(j)})$ to S_{3-j}, $\Delta m_{\rho_2}^{(j)}$ to S_3 and S_4 on behalf of the honest party S_j;
- Upon receiving $(\textsc{CompFetch}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{3,4\}$ from the external $\mathcal{F}_{\mathsf{sot}}^{N,\ell}$, Sim does:
  - Set $\hat{x}_0^{(j)} := r_1 + r_2 \pmod{2^{\ell_1}}$, $\hat{x}_1^{(j)} := r_3 + r_4 \pmod{2^{\ell_1}}$;
  - Set $\Delta m_{\rho_1}^{(j)} := 0$, $\Delta m_{\rho_2}^{(j)} := -w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\tilde{x}_i^{(j)} := -r_{j+2i-2} - w_{x,i}^{(j)} \pmod{2^{\ell_1}}$, $i \in \{0,1\}$;
  - Send $\{\Delta m_{\rho_1}^{(j)}, \tilde{\mathbf{x}}^{(j)}\}$ to S_1 and S_2, $\Delta m_{\rho_2}^{(j)}$ to S_{7-j};
- Upon receiving $(\Delta m_{\rho_1}^{(3)}, \tilde{\mathbf{x}}^{(3)})$ from the corrupted S_3 to S_1, S_2 and $\Delta m_{\rho_2}^{(3)}$ from the corrupted S_3 to S_4, the simulator Sim does:
  - Extract $x_i^{(3)} := \tilde{x}_i^{(4)} + r_{1+2i} + w_{x,i}^{(1)} \pmod{2^{\ell_1}}$, $i \in \{0,1\}$;
  - Extract $\Delta m^{(4)} := \Delta m_{\rho_2}^{(3)} + w_{m,2}^{(1)} \pmod{2^{\ell_2}}$;
  - Pick random $m_0^{(3)}, m_1^{(3)} \leftarrow \mathbb{Z}_{2^{\ell_2}}$ s.t. $m_1^{(3)} - m_0^{(3)} = \Delta m^{(3)}$;
  - Send $(\textsc{CompFetch}, \mathsf{sid}, \mathbf{x}^{(3)}, \mathbf{m}^{(3)})$ to the external $\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2}$;
  - Set $\Delta m_{\rho_2} := \sum_{i=1}^4 \Delta m_{\rho_2}^{(i)} \pmod{2^{\ell_2}}$;
  - Set $\beta_2^{(1)} \leftarrow \mathsf{DCF.Eval}^{\mathsf{IC}}(1, \mathcal{K}_{\rho_2}^{(1)}, \Delta m_{\rho_2})$;
  - Set $\zeta_1 \leftarrow \mathsf{PRF}_{\eta_1}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid}, 3)$, $\zeta_4 \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^{\ell_1}}}(\mathsf{sid}, 5)$;
  - Compute $y^{(4)} := \beta_2^{(1)} \cdot \hat{x}_0^{(3)} + (1 - \beta_2^{(1)}) \cdot \hat{x}_1^{(3)} - \zeta_1 - \zeta_4$;
  - Send $(\textsc{Rand}, \mathsf{sid}, y^{(4)})$ to the external $\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2}$.

Indistinguishability. We assume that the parties $S_1, \ldots, S_4$ 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 $\mathsf{Exec}_{\Pi_{\mathsf{csot}}^{\ell_1,\ell_2},\mathcal{A},\mathcal{Z}}$.

Hybrid $\mathcal{H}_1$: $\mathcal{H}_1$ is the same as $\mathcal{H}_0$ except that in $\mathcal{H}_1$, $\{w_{x,0}^{(j)}\}_{j\in[2]}$, $\{w_{x,1}^{(j)}\}_{j\in[2]}$ and $\{r_j\}_{j\in[4]}$ are picked uniformly random from $\mathbb{Z}_{2^{\ell_1}}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}$; $\{w_{m,2}^{(j)}\}_{j\in[2]}$ and $\rho_2^{(2)}$ are picked uniformly random from $\mathbb{Z}_{2^{\ell_2}}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}$. Set $w_{x,0}^{(3)} := w_{x,0}^{(1)}$, $w_{x,0}^{(4)} := w_{x,0}^{(2)}$, $w_{x,1}^{(3)} := w_{x,1}^{(1)}$, $w_{x,1}^{(4)} := w_{x,1}^{(2)}$, $w_{m,2}^{(3)} := w_{m,2}^{(1)}$, $w_{m,2}^{(4)} := w_{m,2}^{(2)}$.

Claim 8. *If $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A})$, $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^{\ell_2}}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A})$, then $\mathcal{H}_1$ and $\mathcal{H}_0$ are indistinguishable with advantage $\varepsilon_1 := 8 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A})$.*

Proof. We have changed 8 $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}$ outputs and 3 $\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}$ outputs to uniformly random strings; therefore, the overall advantage is $8 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A})$ 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 $j \in \{1,2\}$:
  - Set $\Delta m_{\rho_1}^{(j)} := 0$ and $\Delta m_{\rho_2}^{(j)} := \rho_2^{(j)} + w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;
- For $j \in \{3,4\}$:

– Set $\Delta m_{\rho_1}^{(j)} := 0, \Delta m_{\rho_2}^{(j)} := -w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;

instead of

- For $j \in \{1,2\}$:
  - Set $\Delta m_{\rho_1}^{(j)} := m_1^{(j)} - m_0^{(j)} + w_{m,1}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\Delta m_{\rho_2}^{(j)} := m_1^{(j)} - m_0^{(j)} + \rho_2^{(j)} + w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;
- For $j \in \{3,4\}$:
  - Set $\Delta m_{\rho_1}^{(j)} := m_1^{(j)} - m_0^{(j)} + \rho_1^{(j-2)} - w_{m,1}^{(j)} \pmod{2^{\ell_2}}$;
  - Set $\Delta m_{\rho_2}^{(j)} := m_1^{(j)} - m_0^{(j)} - w_{m,2}^{(j)} \pmod{2^{\ell_2}}$;

Claim 9. *If* $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}} := (\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ *is a secure function secret sharing scheme for offset comparison function* $f_{\alpha,\beta}^{\mathsf{IC}}(x) : \mathbb{Z}_{2^{\ell_2}} \mapsto \mathbb{Z}_{2^{\ell_1}}$ *with adversarial advantage* $\mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A})$*, then* $\mathcal{H}_2$ *and* $\mathcal{H}_1$ *are indistinguishable with advantage* $\varepsilon_2 := \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A})$.

Proof. Note that the corrupted party S_3 only sees $\{\Delta m_{\rho_2}^{(j)}\}_{j\in[4]}$; therefore, the modification of $\{\Delta m_{\rho_1}^{(j)}\}_{j\in[4]}$ is oblivious to S_1. In the hybrid $\mathcal{H}_1$, we have

- $\Delta m_{\rho_2}^{(1)} := m_1^{(1)} - m_0^{(1)} + \rho_2^{(1)} + w_{m,2}^{(1)} \pmod{2^{\ell_2}}$;
- $\Delta m_{\rho_2}^{(2)} := m_1^{(2)} - m_0^{(2)} + \rho_2^{(2)} + w_{m,2}^{(2)} \pmod{2^{\ell_2}}$;
- $\Delta m_{\rho_2}^{(3)} := m_1^{(3)} - m_0^{(3)} - w_{m,2}^{(3)} \pmod{2^{\ell_2}}$;
- $\Delta m_{\rho_2}^{(4)} := m_1^{(4)} - m_0^{(4)} - w_{m,2}^{(4)} \pmod{2^{\ell_2}}$;

It is straightforward that the distribution of $\{\Delta m_{\rho_2}^{(j)}\}_{j\in[4]}$ are uniformly random under the condition $\Delta m_{\rho_2} := \sum_{k=1}^{4} \Delta m_{\rho_2}^{(k)} = m_1 - m_0 + \rho_2$, where ρ_2 is used to generate the DCF keys $\mathcal{K}_{\rho_2}^{(1)}, \mathcal{K}_{\rho_2}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell_2-1}, 1, \mathbb{Z}_{2^{\ell_2}}, \mathbb{Z}_{2^{\ell_1}}, \rho_2)$. Whereas $\Delta m_2 := \rho_2$ 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 $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}} := (\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\varepsilon_2 := \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A})$. □

The adversary's view of $\mathcal{H}_2$ is identical to the simulated view $\mathsf{Exec}_{\mathcal{F}_{\mathsf{csot}}^{\ell_1,\ell_2},\mathcal{S},\mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$8 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_1}}}}(1^\lambda, \mathcal{A}) + 3 \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^{\ell_2}}}}(1^\lambda, \mathcal{A}) + \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^{\ell_2}},\mathbb{Z}_{2^{\ell_1}}}(1^\lambda, \mathcal{A}).$$

This concludes the proof. □

C Proof of Theorem 3

Theorem 3. Let $\mathsf{DPF}_{\mathsf{IC}}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}$ be a secure function secret sharing scheme for offset comparison function $f_{\alpha,\beta}(x) : \mathbb{Z}_{2^\ell} \mapsto \mathbb{Z}_{2^\lambda}$ with adversarial advantage $\mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}(1^\lambda, \mathcal{A})$. 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 $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\lambda}$ be a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$. The protocol $\Pi_{\mathsf{os}}^{\mathsf{const}}$ as described in Fig. 11 and $\Pi_{\mathsf{eval}}^{\mathsf{const}}$ as described in Fig. 12 UC-realizes $\mathcal{F}_{\mathsf{bp}}^4$ 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

$$3m_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + (m_c + 1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A}) + m_c \cdot \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}(1^\lambda, \mathcal{A})$$

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, $\mathcal{S} := \{S_1, \ldots, S_4\}$ 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 $\mathsf{Exec}_{\mathcal{F}_{\mathsf{bp}}^4,\mathsf{Sim},\mathcal{Z}}$ where the parties $M, D, S_1, \ldots, S_4$ interact with functionality $\mathcal{F}_{\mathsf{bp}}^4$ in the ideal world, and corrupted parties are controlled by the simulator Sim. We consider following cases.

Case 1: S_1 (or 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_2, S_3, S_4. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon receiving $(\textsc{Model}, \mathsf{sid}, M, (m,d))$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, the simulator Sim computes $m_c := 2^{d-1} - 1$ and acts as the honest model owner M to do:
  - **for** $i := 0$ to $m_c - 1$:
    - Set $k_i^{(1)}, k_i^{(2)} \leftarrow \mathbb{Z}_n$;
    - Set $t_i^{(1)}, t_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - Set $P_i^{(1)} := \{k_i^{(1)}, t_i^{(1)}\}$, $P_i^{(2)} := \{k_i^{(2)}, t_i^{(2)}\}$;
  - **for** $i := 0$ to m_c:
    - $v_i^{(1)}, v_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$;
  - Send $(\mathcal{P}^{(1)}, \mathbf{v}^{(1)})$ to S_1, $(\mathcal{P}^{(2)}, \mathbf{v}^{(2)})$ to S_2.
- Upon receiving $(\textsc{Data}, \mathsf{sid}, D, n)$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, the simulator Sim acts as the honest data owner D to do:
  - **for** $i := 0$ to $n - 1$ do:
    - Generate $x_i^{(1)}, x_i^{(3)} \leftarrow \mathbb{Z}_{2^\ell}$, set $x_i^{(2)} := x_i^{(1)}$, $x_i^{(4)} := x_i^{(3)}$;
  - Send $\mathbf{x}^{(j)}$ to S_j, $j \in [4]$.
- Upon initialization, the simulator Sim acts as the honest party S_3 to do:
  - **for** $i := 0$ to $m_c - 1$:
    - Generate $\rho_i, \rho_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$ and set $\rho_i^{(1)} := \rho_i - \rho_i^{(2)}$;

* $\mathcal{K}_{i,\rho}^{(1)}, \mathcal{K}_{i,\rho}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell-1}, 1, \mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}, \rho_i)$;
  - Send $(\mathcal{K}_{i,\rho}^{(1)})_{i\in\mathbb{Z}_{m_c}}$ to S_1, $(\mathcal{K}_{i,\rho}^{(2)})_{i\in\mathbb{Z}_{m_c}}$ to S_2;
- The simulator Sim does:
  - For $i \in \mathbb{Z}_{m_c}$, pick random $w_i^{(1)}, w_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$ and $r_i \leftarrow \mathbb{Z}_{2^\lambda}$;
  - Pick random $\delta \leftarrow \mathbb{Z}_{m_c+1}$
  - For $i \in \mathbb{Z}_{m_c}$, set $w_i^{(3)} := w_i^{(1)}$ and $w_i^{(4)} := w_i^{(2)}$;
- Upon receiving $(\mathsf{Eval}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1,2\}$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, Sim does:
  - **for** $i := 0$ to $m_c - 1$:
    * Set $\tilde{k}_i^{(j)} := 0$;
    * Send $(\text{FETCH}, \mathsf{sid}, \mathbf{x}^j, \tilde{k}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    * Set $\Delta x_i^{(j)} := w_i^{(j)} - x_{k_i}^{(j)} \pmod{2^\ell}$;
  - Send $\Delta\mathbf{x}^{(j)}$ to S_{3-j} on behalf of the honest party S_j;
- Upon receiving $(\mathsf{Eval}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{3,4\}$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, Sim does:
  - **for** $i := 0$ to $m_c - 1$:
    * Set $\tilde{k}_i^{(j)} := 0$;
    * Send $(\text{FETCH}, \mathsf{sid}, \mathbf{x}^j, \tilde{k}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    * Set $\Delta x_i^{(j)} := \rho_i^{(j-2)} - x_{k_i}^{(j)} - w_i^{(j)} \pmod{2^\ell}$;
  - $\Delta\mathbf{x}^{(j)}$ to S_1 and S_2 on behalf of the honest party S_j;
- Upon receiving $\Delta\mathbf{x}^{(3-j)}$ from S_{3-j}, $\Delta\mathbf{x}^{(3)}$ from S_3 , and $\Delta\mathbf{x}^{(4)}$ from S_4, for an honest party S_j, $j \in \{1,2\}$, Sim does:
  - **for** $i := 0$ to $m_c - 1$:
    * Set $\Delta x_i := \sum_{q=1}^4 \Delta x_i^{(q)} \pmod{2^\ell}$;
    * Set $b_i^{(j)} \leftarrow \mathsf{DCF.Eval}^{\mathsf{IC}}(j, \mathcal{K}_{i,\rho}^{(j)}, \Delta x_i)$;
    * Set $e_{i,1}^{(j)} := (j - 1 - b_i^{(j)}) \cdot r_i \pmod{2^\lambda}$;
    * Set $e_{i,2}^{(j)} := b_i^{(j)} \cdot r_i \pmod{2^\lambda}$;
  - **for** $i := 0$ to m_c :
    * Sum up the share of edge costs along i-th leaf node's path to get $c_i^{(j)}$, set $\hat{c}_i^{(j)} := c_{i-\delta \pmod{m_c+1}}^{(j)}$;
    * Set $\hat{v}_{i,1}^{(j)} \leftarrow \mathbb{Z}_{2^\ell}$, $\hat{v}_{i,2}^{(j)} := v_{i-\delta \pmod{m_c+1}}^{(j)} - \hat{v}_{i,1}^{(j)} \pmod{2^\ell}$;
  - Send $(\hat{c}_i^{(j)}, \hat{v}_{i,1}^{(j)})_{i\in\mathbb{Z}_{m_c+1}}$ to S_3, $(\hat{c}_i^{(j)}, \hat{v}_{i,2}^{(j)})_{i\in\mathbb{Z}_{m_c+1}}$ to S_4;

Indistinguishability. We assume that the parties $S_1, \ldots, S_4$ 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 $\mathcal{H}_1$, $\{w_i^{(1)}\}_{i\in\mathbb{Z}_{m_c}}$, $\{w_i^{(2)}\}_{i\in\mathbb{Z}_{m_c}}$ and $\{\rho_i^{(2)}\}_{i\in\mathbb{Z}_{m_c}}$ are picked uniformly random from $\mathbb{Z}_{2^\ell}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}$; $\{r_i\}_{i\in\mathbb{Z}_{m_c+1}}$ is picked uniformly random from $\mathbb{Z}_{2^\lambda}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}$. Set $w_i^{(3)} := w_i^{(1)}$, $w_i^{(4)} := w_i^{(2)}$, $i \in \mathbb{Z}_{m_c}$.

Claim 10. *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* $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A})$*,* $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\lambda}$ *is a secure pseudorandom function with adversarial advantage* $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$*, then $\mathcal{H}_1$ and $\mathcal{H}_0$ are indistinguishable with advantage* $\varepsilon_1 := 3m_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + (m_c+1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$.

Proof. We have changed $3m_c$ $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}$ outputs and (m_c+1) $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}$ outputs to uniformly random strings; therefore, the overall advantage is $2m_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda, \mathcal{A}) + (m_c+1) \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda, \mathcal{A})$ 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 $j \in \{1,2\}$,:
- Set $\Delta x_i^{(j)} := w_i^{(j)} - x_{k_i}^{(j)} \pmod{2^\ell}$, $i \in \mathbb{Z}_{m_c}$;

instead of

For $j \in \{1,2\}$:
- Set $\Delta x_i^{(j)} := t_i^{(j)} - x_{k_i}^{(j)} + w_i^{(j)} \pmod{2^\ell}$, $i \in \mathbb{Z}_{m_c}$;

Claim 11. *If* $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}} := (\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ *is a secure function secret sharing scheme for offset comparison function* $f_{\alpha,\beta}^{\mathsf{IC}}(x) : \mathbb{Z}_{2^\ell} \mapsto \mathbb{Z}_{2^\lambda}$ *with adversarial advantage* $\mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}}(1^\lambda, \mathcal{A})$*, then $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with advantage* $\varepsilon_2 := m_c \cdot \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}}(1^\lambda, \mathcal{A})$.

Proof. Note that the corrupted party S_1 only sees $\{\Delta m_{\rho_1}^{(j)}\}_{j\in[4]}$; therefore, the modification of $\{\Delta m_{\rho_2}^{(j)}\}_{j\in[4]}$ is oblivious to S_1. In the hybrid $\mathcal{H}_1$, we have $i \in \mathbb{Z}_{m_c}$
- $\Delta x_i^{(1)} := \Delta x_i^{(1)} := t_i^{(1)} - x_{k_i}^{(1)} + w_i^{(1)} \pmod{2^\ell}$;
- $\Delta x_i^{(2)} := \Delta x_i^{(j)} := t_i^{(2)} - x_{k_i}^{(2)} + w_i^{(2)} \pmod{2^\ell}$;
- $\Delta x_i^{(3)} := \rho_i^{(1)} - x_{k_i}^{(3)} - w_i^{(3)} \pmod{2^\ell}$;
- $\Delta x_i^{(4)} := \rho_i^{(2)} - x_{k_i}^{(4)} - w_i^{(4)} \pmod{2^\ell}$;

It is straightforward that for $i \in \mathbb{Z}_{m_c}$, the distribution of $\{\Delta x_i^{(j)}\}_{j\in[4]}$ are uniformly random under the condition $\Delta x_i := \sum_{q=1}^4 \Delta x_i^{(q)} = t_i - x_{k_i} + \rho_1$, where ρ_i is used to generate the DCF keys $\mathcal{K}_{i,\rho}^{(1)}, \mathcal{K}_{i,\rho}^{(2)} \leftarrow \mathsf{DCF.Gen}^{\mathsf{IC}}(1^\lambda, 2^{\ell-1}, 1, \mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}, \rho_i)$. Whereas $\Delta x_i := \rho_i - x_{k_i}$ 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 $\mathsf{DCF}_{\mathsf{IC}}^{\mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}} := (\mathsf{Gen}^{\mathsf{IC}}, \mathsf{Eval}^{\mathsf{IC}})$ with the same advantage. Therefore, $\mathcal{H}_2$ and $\mathcal{H}_1$ are indistinguishable with adversarial advantage $\varepsilon_2 := m_c \cdot \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell}, \mathbb{Z}_{2^\lambda}}(1^\lambda, \mathcal{A})$. □

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

For $j \in \{1,2\}$:
- Set $\tilde{k}_i^{(j)} := 0$, $i \in \mathbb{Z}_{m_c}$;

  instead of

For $j \in \{1,2\}$:

- Set $\tilde{k}_i^{(j)} := k_i^{(j)}$, $i \in \mathbb{Z}_{m_c}$;

Claim 12. *$\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable.*

Proof. Since $\{\tilde{k}_i^{(j)}\}_{i\in\mathbb{Z}_{m_c}, j\in[2]}$ are only sent to the simulated $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$, which is oblivious to S_1, $\mathcal{H}_3$ and $\mathcal{H}_2$ are perfectly indistinguishable. □

The adversary's view of $\mathcal{H}_3$ is identical to the simulated view $\mathsf{Exec}_{\mathcal{F}_{\mathsf{bp}}^4,\mathcal{S},\mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$3m_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda,\mathcal{A}) + (m_c+1)\cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda,\mathcal{A}) + m_c \cdot \mathsf{Adv}_{\mathsf{DCF}_{\mathsf{IC}}}^{\mathbb{Z}_{2^\ell},\mathbb{Z}_{2^\lambda}}(1^\lambda,\mathcal{A})$$

Case 2: S_3 (or S_4) 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_2, S_3, S_4. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon receiving $(\textsc{Model}, \mathsf{sid}, M, (m,d))$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, the simulator Sim computes $m_c := 2^{d-1}-1$ and acts as the honest model owner M to do:
  - **for** $i := 0$ to $m_c - 1$:
    - Set $k_i^{(1)}, k_i^{(2)} \leftarrow \mathbb{Z}_n$;
    - Set $t_i^{(1)}, t_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$;
    - Set $P_i^{(1)} := \{k_i^{(1)}, t_i^{(1)}\}$, $P_i^{(2)} := \{k_i^{(2)}, t_i^{(2)}\}$;
  - **for** $i := 0$ to m_c:
    - $v_i^{(1)}, v_i^{(2)} \leftarrow \mathbb{Z}_{2^\ell}$;
  - Send $(\mathcal{P}^{(1)}, \mathbf{v}^{(1)})$ to S_1, $(\mathcal{P}^{(2)}, \mathbf{v}^{(2)})$ to S_2.
- Upon receiving $(\textsc{Data}, \mathsf{sid}, D, n)$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, the simulator Sim acts as the honest data owner D to do:
  - **for** $i := 0$ to $n-1$ do:
    - Generate $x_i^{(1)}, x_i^{(3)} \leftarrow \mathbb{Z}_{2^\ell}$, set $x_i^{(2)} := x_i^{(1)}$, $x_i^{(4)} := x_i^{(3)}$;
  - Send $\mathbf{x}^{(j)}$ to S_j, $j \in [4]$.
- Upon receiving $(\mathsf{Eval}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{1,2\}$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, Sim does:
  - **for** $i := 0$ to $m_c - 1$:
    - Set $\tilde{k}_i^{(j)} := 0$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^j, \tilde{k}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    - Set $\Delta x_i^{(j)} := w_i^{(j)} - x_{k_i}^{(j)} \pmod{2^\ell}$;
  - Send $\Delta\mathbf{x}^{(j)}$ to S_{3-j} on behalf of the honest party S_j;
- Upon receiving $(\mathsf{Eval}, \mathsf{sid}, S_j)$ for an honest party S_j, $j \in \{3,4\}$ from the external $\mathcal{F}_{\mathsf{bp}}^4$, Sim does:
  - **for** $i := 0$ to $m_c - 1$:
    - Set $\tilde{k}_i^{(j)} := 0$;
    - Send $(\textsc{Fetch}, \mathsf{sid}, \mathbf{x}^j, \tilde{k}_i^{(j)})$ to $\mathcal{F}_{\mathsf{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    - Set $\Delta x_i^{(j)} := \rho_i^{(j-2)} - x_{k_i}^{(j)} - w_i^{(j)} \pmod{2^\ell}$;
  - $\Delta\mathbf{x}^{(j)}$ to S_1 and S_2 on behalf of the honest party S_j;
- Upon receiving $\Delta\mathbf{x}^{(3-j)}$ from S_{3-j}, $((\mathcal{K}_{i,\rho}^{(j)})_{i\in\mathbb{Z}_{m_c}}, \Delta\mathbf{x}^{(3)})$ from S_3, and $\Delta\mathbf{x}^{(4)}$ from S_4, for an honest party S_j, $j \in \{1,2\}$, Sim does:
  - **for** $i := 0$ to $m_c - 1$:
    - Set $\Delta x_i := \sum_{q=1}^4 \Delta x_i^{(q)} \pmod{2^\ell}$;
    - Set $b_i^{(j)} \leftarrow \mathsf{DCF.Eval}^{\mathsf{IC}}(j, \mathcal{K}_{i,\rho}^{(j)}, \Delta x_i)$;
    - Set $e_{i,1}^{(j)} := (j - 1 - b_i^{(j)}) \cdot r_i \pmod{2^\lambda}$;
    - Set $e_{i,2}^{(j)} := b_i^{(j)} \cdot r_i \pmod{2^\lambda}$;
  - **for** $i := 0$ to m_c :
    - Sum up the share of edge costs along i-th leaf node's path to get $c_i^{(j)}$, set $\hat{c}_i^{(j)} := c_{i-\delta \pmod{m_c+1}}^{(j)}$;
    - Set $\hat{v}_{i,1}^{(j)} \leftarrow \mathbb{Z}_{2^\ell}$, $\hat{v}_{i,2}^{(j)} := v_{i-\delta \pmod{m_c+1}}^{(j)} - \hat{v}_{i,1}^{(j)} \pmod{2^\ell}$;
  - Send $(\hat{c}_i^{(j)}, \hat{v}_{i,1}^{(j)})_{i\in\mathbb{Z}_{m_c+1}}$ to S_3, $(\hat{c}_i^{(j)}, \hat{v}_{i,2}^{(j)})_{i\in\mathbb{Z}_{m_c+1}}$ to S_4;

Indistinguishability. We assume that the parties $S_1,\dots,S_4$ 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 $\mathcal{H}_0,\dots,\mathcal{H}_3$.

Hybrid $\mathcal{H}_0$: It is the real protocol 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 $\mathcal{H}_1$, $\{w_i^{(1)}\}_{i\in\mathbb{Z}_{m_c}}$, $\{w_i^{(2)}\}_{i\in\mathbb{Z}_{m_c}}$ and $\{\rho_i^{(2)}\}_{i\in\mathbb{Z}_{m_c}}$ are picked uniformly random from $\mathbb{Z}_{2^\ell}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}$; $\{r_i\}_{i\in\mathbb{Z}_{m_c+1}}$ is picked uniformly random from $\mathbb{Z}_{2^\lambda}$ instead of calculating from $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}$. Set $w_i^{(3)} := w_i^{(1)}$, $w_i^{(4)} := w_i^{(2)}$, $i \in \mathbb{Z}_{m_c}$.

Claim 13. *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 $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda,\mathcal{A})$, $\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}} : \{0,1\}^\lambda \times \{0,1\}^{\mathsf{in}} \mapsto \mathbb{Z}_{2^\lambda}$ is a secure pseudorandom function with adversarial advantage $\mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda,\mathcal{A})$, then $\mathcal{H}_1$ and $\mathcal{H}_0$ are indistinguishable with advantage $\varepsilon_1 := 3m_c \cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda,\mathcal{A}) + (m_c+1)\cdot \mathsf{Adv}_{\mathsf{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda,\mathcal{A})$.*

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

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

For $j \in \{1,2\}$:

- Set $\tilde{k}_i^{(j)} := 0$, $i \in \mathbb{Z}_{m_c}$;
  instead of

For $j \in \{1,2\}$:

- Set $\tilde{k}_i^{(j)} := k_i^{(j)}$, $i \in \mathbb{Z}_{m_c}$;

Claim 14. *$\mathcal{H}_2$ and $\mathcal{H}_1$ are perfectly indistinguishable.*

Proof. Since $\{\tilde{k}_i^{(j)}\}_{i\in\mathbb{Z}_{m_c},j\in[2]}$ are only sent to the simulated $\mathcal{F}_{\text{sot}}^{n,\ell}$, which is oblivious to S_1, $\mathcal{H}_2$ and $\mathcal{H}_1$ are perfectly indistinguishable. □

The adversary's view of $\mathcal{H}_2$ is identical to the simulated view $\text{Exec}_{\mathcal{F}_{\text{bp}}^4,\mathcal{S},\mathcal{Z}}$. Therefore, the overall distinguishing advantage is

$$3m_c \cdot \text{Adv}_{\text{PRF}^{\mathbb{Z}_{2^\ell}}}(1^\lambda,\mathcal{A}) + (m_c+1)\cdot \text{Adv}_{\text{PRF}^{\mathbb{Z}_{2^\lambda}}}(1^\lambda,\mathcal{A})$$

This concludes the proof. □

D Proof of Theorem 4

Theorem 4. The protocol $\Pi_{\text{os}}^{\text{poly}}$ as described in Fig. 15 and $\Pi_{\text{eval}}^{\text{poly}}$ as described in Fig. 16 UC-realizes $\mathcal{F}_{\text{bp}}^4$ as described in Fig. 2 in the $\{\mathcal{F}_{\text{sot}},\mathcal{F}_{\text{csot}}\}$-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 $\text{Exec}^{\mathcal{F}_{\text{sot}},\mathcal{F}_{\text{csot}}}_{\{\Pi_{\text{os}}^{\text{poly}},\Pi_{\text{eval}}^{\text{poly}}\},\mathcal{A},\mathcal{Z}}$ where the parties M,D, $\mathcal{S} := \{S_1,\dots,S_4\}$ run protocol $\Pi_{\text{os}}^{\text{poly}}$, $\Pi_{\text{eval}}^{\text{poly}}$ in the $\{\mathcal{F}_{\text{sot}},\mathcal{F}_{\text{csot}}\}$-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 $\text{Exec}_{\mathcal{F}_{\text{bp}}^4,\text{Sim},\mathcal{Z}}$ where the parties $M,D,S_1,\dots,S_4$ interact with functionality $\mathcal{F}_{\text{bp}}^4$ in the ideal world, and corrupted parties are controlled by the simulator Sim.

Simulator. The simulator Sim internally runs $\mathcal{A}$, forwarding messages to/from the environment $\mathcal{Z}$. Sim simulates the interface of $\{\mathcal{F}_{\text{sot}},\mathcal{F}_{\text{csot}}\}$ as well as honest parties M,D, S_2,S_3,S_4. In addition, the simulator Sim simulates the following interactions with $\mathcal{A}$.

- Upon receiving $(\textsc{Model},\text{sid},M,(m,d))$ from the external $\mathcal{F}_{\text{bp}}^4$, the simulator Sim acts as the honest model owner M to do:
  - Build the position mapping, using dummy elements as $P_i := \{I_i^{\text{left}} := 0, I_i^{\text{right}} := 0, \mathcal{J}_i^{\text{left}} := 0, \mathcal{J}_i^{\text{right}} := 0, t_i := 0, v_i := 0\}$;
  - **for** $i := 0$ to $m-1$ do:
    - ∗ Set $I_i^{\text{left},(1)} \leftarrow \mathbb{Z}_m$, $I_i^{\text{left},(2)} := I_i^{\text{left},(1)}$;
    - ∗ Set $I_i^{\text{left},(3)} = I_i^{\text{left},(4)} := I_i^{\text{left}} - I_i^{\text{left},(1)} \pmod m$;
    - ∗ Set $\mathcal{J}_i^{\text{left},(1)} \leftarrow \mathbb{Z}_n$, $\mathcal{J}_i^{\text{left},(2)} := \mathcal{J}_i^{\text{left},(1)}$;
    - ∗ Set $\mathcal{J}_i^{\text{left},(3)} = \mathcal{J}_i^{\text{left},(4)} := \mathcal{J}_i^{\text{left}} - \mathcal{J}_i^{\text{left},(1)} \pmod n$;
    - ∗ Set $I_i^{\text{right},(1)} \leftarrow \mathbb{Z}_m$, $I_i^{\text{right},(2)} := I_i^{\text{right},(1)}$;
    - ∗ Set $I_i^{\text{right},(3)} = I_i^{\text{right},(4)} := I_i^{\text{right}} - I_i^{\text{right},(1)} \pmod m$;
    - ∗ Set $\mathcal{J}_i^{\text{right},(1)} \leftarrow \mathbb{Z}_n$, $\mathcal{J}_i^{\text{right},(2)} := \mathcal{J}_i^{\text{right},(1)}$;
    - ∗ Set $\mathcal{J}_i^{\text{right},(3)} = \mathcal{J}_i^{\text{right},(4)} := \mathcal{J}_i^{\text{right}} - \mathcal{J}_i^{\text{right},(1)} \pmod n$;
    - ∗ Set $t_i^{(1)} \leftarrow \mathbb{Z}_{2^\lambda}$, $t_i^{(2)} := t_i^{(1)}$;
    - ∗ Set $t_i^{(3)} = t_i^{(4)} := t_i - t_i^{(1)} \pmod{2^\lambda}$;
    - ∗ Set $v_i^{(1)} \leftarrow \mathbb{Z}_{2^\lambda}$, $v_i^{(2)} := v_i^{(1)}$;
    - ∗ Set $v_i^{(3)} = v_i^{(4)} := v_i - v_i^{(1)} \pmod{2^\lambda}$;
  - Set $\text{id}_1 := 1$ and $k_1 := 0$;
  - Generate $\text{id}_1^{(1)},\dots,\text{id}_1^{(4)} \leftarrow \mathbb{Z}_m$, $\text{id}_1 = \sum_{i=1}^4 \text{id}_1^{(i)} \pmod m$;
  - Generate $k_1^{(1)},k_1^{(2)},k_1^{(3)},k_1^{(4)} \leftarrow \mathbb{Z}_n$, $k_1 = \sum_{i=1}^4 k_1^{(i)} \pmod n$;
  - Send $(\mathcal{P}^{(j)},\text{id}_1^{(j)},k_1^{(j)})$ to S_j, $j\in[4]$;
- Upon receiving $(\textsc{Data},\text{sid},D,n)$ from the external $\mathcal{F}_{\text{bp}}^4$, the simulator Sim acts as the honest data owner D to do:
  - **for** $i := 0$ to $n-1$ do:
    - ∗ Generate $x_i^{(1)},x_i^{(3)} \leftarrow \mathbb{Z}_{2^\ell}$, set $x_i^{(2)} := x_i^{(1)}$, $x_i^{(4)} := x_i^{(3)}$;
  - Send $\mathbf{x}^{(j)}$ to S_j, $j\in[4]$.
- Upon receiving $(\text{Eval},\text{sid},S_j)$ for an honest party S_j, from the external $\mathcal{F}_{\text{bp}}^4$, Sim does:
  - **for** $i := 1$ to d:
    - ∗ Send $(\textsc{Fetch},\text{sid},\mathbf{x}^{(j)},k_i^{(j)})$ to $\mathcal{F}_{\text{sot}}^{n,\ell}$ to get $x_{k_i}^{(j)}$;
    - ∗ Send $(\textsc{Fetch},\text{sid},\mathcal{P}^{(j)},\text{id}_i^{(j)})$ to $\mathcal{F}_{\text{sot}}^{\widetilde{m},*}$ to get $P_{id_i}^{(j)} := (I_{id_i}^{\text{left},(j)}, I_{id_i}^{\text{right},(j)}, \mathcal{J}_{id_i}^{\text{left},(j)}, \mathcal{J}_{id_i}^{\text{right},(j)}, t_{id_i}^{(j)}, v_{id_i}^{(j)})$;
    - ∗ Set $\text{res}^{(j)} := \text{res}^{(j)} + v_{id_i}^{(j)} \pmod{2^\ell}$;
    - ∗ **if** $i \geq d$, return $\text{res}^{(j)}$ to the receiver R and **break**;
    - ∗ Obliviously fetch $\text{id}_{i+1}^{(j)}$ and $k_{i+1}^{(j)}$:
      - · Send $(\textsc{CompFetch},\text{sid},(I_{id_i}^{\text{left},(j)}, I_{id_i}^{\text{right},(j)}),(x_{k_i}^{(j)},t_{id_i}^{(j)}))$ to $\mathcal{F}_{\text{csot}}^{\log\widetilde{m},\ell}$
      - · Send $(\textsc{CompFetch},\text{sid},(\mathcal{J}_{id_i}^{\text{left},(j)}, \mathcal{J}_{id_i}^{\text{right},(j)}),(x_{k_i}^{(j)},t_{id_i}^{(j)}))$ to $\mathcal{F}_{\text{csot}}^{\log n,\ell}$
- When the simulated $\{\mathcal{F}_{\text{sot}},\mathcal{F}_{\text{csot}}\}$ receives input from the corrupted party S_j, the simulator Sim sends (Eval,sid) to the external $\mathcal{F}_{\text{bp}}^4$;
- When the simulated receiver R terminates, the simulator Sim allows the $(\textsc{Result},\text{sid},y)$ message to be delivered to R in the ideal world.

Indistinguishability. We assume that the parties M,D, $S_1,\dots,S_4$ communicate with each other via the secure channel functionality $\mathcal{F}_{\text{sc}}$ (omitted in the protocol description for simplicity). The views of $\mathcal{A}$ and $\mathcal{Z}$ in $\text{Exec}^{\mathcal{F}_{\text{sot}},\mathcal{F}_{\text{csot}}}_{\{\Pi_{\text{os}}^{\text{poly}},\Pi_{\text{eval}}^{\text{poly}}\},\mathcal{A},\mathcal{Z}}$ and $\text{Exec}_{\mathcal{F}_{\text{bp}}^4,\text{Sim},\mathcal{Z}}$ are identical. Therefore, it is perfectly indistinguishable. This concludes the proof. □

E Oblivious Selection Π_{sel}

In this section, we provide the details of our oblivious selection protocol invoked in our polynomial-round private decision tree protocol $\Pi_{\text{eval}}^{\text{poly}}$ in Sec. 6.2. The protocol Π_{sel} is described in Fig. 18.

Selection Protocol Π_{sel}

Offline phase:

- S_1 dose for $i = 1$ to d:
  - Generate $\varphi_{i,2} \leftarrow \mathbb{Z}_n$, set $\varphi_{i,2}^{(1)} := \varphi_{i,2} - \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_N}(\mathsf{sid}, i)$ and $\mathcal{K}_{i,\varphi_2}^{(1)}, \mathcal{K}_{i,\varphi_2}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \varphi_{i,2}, 1, \mathbb{Z}_n, \mathbb{Z}_{2^\ell})$;
  - Generate $\psi_{i,2} \leftarrow \mathbb{Z}_{\widetilde{m}}$, set $\psi_{i,2}^{(1)} := \psi_{i,2}$ and $\mathcal{K}_{i,\psi_2}^{(1)}, \mathcal{K}_{i,\psi_2}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \psi_{i,2}, 1, \mathbb{Z}_{\widetilde{m}}, \mathbb{Z})$;
- S_3 dose for $i = 1$ to d:
  - Generate $\varphi_{i,1} \leftarrow \mathbb{Z}_n$, set $\varphi_{i,1}^{(1)} := \varphi_{i,1} - \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_N}(\mathsf{sid}, i)$ and $\mathcal{K}_{i,\varphi_1}^{(1)}, \mathcal{K}_{i,\varphi_1}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \varphi_{i,1}, 1, \mathbb{Z}_n, \mathbb{Z}_{2^\ell})$;
  - Generate $\psi_{i,1} \leftarrow \mathbb{Z}_{\widetilde{m}}$, set $\psi_{i,1}^{(1)} := \psi_{i,1}$ and $\mathcal{K}_{i,\psi_1}^{(1)}, \mathcal{K}_{i,\psi_1}^{(2)} \leftarrow \mathsf{DPF.Gen}(1^\lambda, \psi_{i,1}, 1, \mathbb{Z}_{\widetilde{m}}, \mathbb{Z})$;
- S_1 sends $(Key_{i,\varphi_2}^{(1)}, \mathcal{K}_{i,\psi_2}^{(1)})_{i\in[d]}$ to S_3,$(Key_{i,\varphi_2}^{(2)}, \mathcal{K}_{i,\psi_2}^{(2)})_{i\in[d]}$ to S_4, and S_3 sends $(Key_{i,\varphi_1}^{(1)}, \mathcal{K}_{i,\psi_1}^{(1)})_{i\in[d]}$ to S_1,$(Key_{i,\varphi_1}^{(2)}, \mathcal{K}_{i,\psi_1}^{(2)})_{i\in[d]}$ to S_4;
- S_2 sets $\varphi_{i,2}^{(1)} := \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_N}(\mathsf{sid}, i)$ and $\psi_{i,1}^{(1)} := -\mathsf{PRF}_{\eta_4}^{\mathbb{Z}_N}(\mathsf{sid}, 0)$, S_3 sets $\varphi_{i,2}^{(1)} := 0$ and $\psi_{i,2}^{(1)} := 0$, for $i \in [d]$;

Online phase:

- For select x_{k_i} and P_{id_i}:
  - Upon receiving $(\text{FETCH}, \mathsf{sid}, \mathbf{x}^{(j)}, k_i^{(j)})$ from the environment $\mathcal{Z}$, player $S_j, j \in \{1,2\}$ does:
    * Set $w_{k,1}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_n}(\mathsf{sid},1)$, $w_{k,2}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_n}(\mathsf{sid},2)$, $w_{\mathsf{id},1}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},3)$, $w_{\mathsf{id},2}^{(j)} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},4)$;
    * Set $\delta_{k,1}^{(j)} := k_i^{(j)} + w_{k,1}^{(j)} \pmod n$, $\delta_{k,2}^{(j)} := k_i^{(j)} - \varphi_{i,2}^{(j)} + w_{k,2}^{(j)} \pmod n$;
    * Set $\delta_{\mathsf{id},1}^{(j)} := \mathsf{id}_i^{(j)} + w_{\mathsf{id},1}^{(j)} \pmod{\widetilde{m}}$, $\delta_{\mathsf{id},2}^{(j)} := \mathsf{id}_i^{(j)} - \psi_{i,2}^{(j)} + w_{\mathsf{id},2}^{(j)} \pmod{\widetilde{m}}$;
    * Send $(\delta_{k,1}^{(j)}, \delta_{\mathsf{id},1}^{(j)})$ to S_{3-j}, $(\delta_{k,2}^{(j)}, \delta_{\mathsf{id},2}^{(j)})$ to S_3 and S_4;
  - Upon receiving $(\delta_{k,1}^{(3-j)}, \delta_{\mathsf{id},1}^{(3-j)})$ from S_{3-j}, $(\delta_{k,1}^{(3)}, \delta_{\mathsf{id},1}^{(3)})$ from S_3 , and $(\delta_{k,1}^{(4)}, \delta_{\mathsf{id},1}^{(4)})$ from S_4, player $S_j, j \in \{1,2\}$ does:
    * Set $\delta_{k,1} := \delta_{k,1}^{(1)} + \delta_{k,1}^{(2)} + \delta_{k,1}^{(3)} + \delta_{k,1}^{(4)} \pmod n$, $\widetilde{x}_q^{(j)} := x_{q-\delta_{k,1} \pmod n}^{(j)}$, for $q \in \mathbb{Z}_n$;
    * Set $\delta_{\mathsf{id},1} := \delta_{\mathsf{id},1}^{(1)} + \delta_{\mathsf{id},1}^{(2)} + \delta_{\mathsf{id},1}^{(3)} + \delta_{\mathsf{id},1}^{(4)} \pmod{\widetilde{m}}$, $\widetilde{P}_q^{(j)} := P_{q-\delta_{\mathsf{id},1} \pmod{\widetilde{m}}}^{(j)}$, for $q \in \mathbb{Z}_{\widetilde{m}}$;
    * Set $\zeta_{x,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},1)$, $\zeta_{x,3} \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},1)$, $\zeta_{t,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},2)$, $\zeta_{t,3} \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},2)$, $\zeta_{v,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},3)$, $\zeta_{v,3} \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},3)$, $\zeta_{I_1,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},1)$, $\zeta_{I_1,3} \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},1)$, $\zeta_{I_2,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},2)$, $\zeta_{I_2,3} \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},2)$, $\zeta_{\mathcal{J}_1,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_n}(\mathsf{sid},1)$, $\zeta_{\mathcal{J}_1,3} \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_n}(\mathsf{sid},1)$, $\zeta_{\mathcal{J}_2,j} \leftarrow \mathsf{PRF}_{\eta_j}^{\mathbb{Z}_n}(\mathsf{sid},2)$, $\zeta_{\mathcal{J}_2,3} \leftarrow \mathsf{PRF}_{\eta_3}^{\mathbb{Z}_n}(\mathsf{sid},2)$;
    * Set $(\beta_{q,\varphi_1}^{(j)})_{q\in\mathbb{Z}_n} \leftarrow \mathsf{DPF.EvalAll}(j, \mathcal{K}_{i,\varphi_1}^{(j)})$ and $(\beta_{q,\phi_1}^{(j)})_{q\in\mathbb{Z}_{\widetilde{m}}} \leftarrow \mathsf{DPF.EvalAll}(j, \mathcal{K}_{i,\phi_1}^{(j)})$;
    * Set $x_{k_i}^{(j)} := \sum_{q=0}^{n-1}(\widetilde{x}_q^{(j)} \cdot \beta_{k,\varphi_1}^{(j)}) + \zeta_{x,j} + (-1)^j \cdot \zeta_{x,3}$;
    * Set $I_{\mathsf{id}_i}^{left,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),left} \cdot \beta_{q,\phi_1}^{(j)}) + \zeta_{I_1,j} + (-1)^j \cdot \zeta_{I_1,3}$, $I_{\mathsf{id}_i}^{right,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),right} \cdot \beta_{q,\phi_1}^{(j)}) + \zeta_{I_2,j} + (-1)^j \cdot \zeta_{I_2,3}$;
    * Set $\mathcal{J}_{\mathsf{id}_i}^{left,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),left} \cdot \beta_{q,\phi_1}^{(j)}) + \zeta_{\mathcal{J}_1,j} + (-1)^j \cdot \zeta_{\mathcal{J}_1,3}$, $\mathcal{J}_{\mathsf{id}_i}^{right,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),right} \cdot \beta_{q,\phi_1}^{(j)}) + \zeta_{\mathcal{J}_2,j} + (-1)^j \cdot \zeta_{\mathcal{J}_2,3}$;
    * Set $t_{\mathsf{id}_i}^{(j)} := \sum_{q=0}^{n-1}(\widetilde{t}_q^{(j)} \cdot \beta_{q,\phi_1}^{(j)}) + \zeta_{t,j} + (-1)^j \cdot \zeta_{t,3}$, $v_{\mathsf{id}_i}^{(j)} := \sum_{q=0}^{n-1}(\widetilde{v}_q^{(j)} \cdot \beta_{q,\phi_1}^{(j)}) + \zeta_{v,j} + (-1)^j \cdot \zeta_{v,3}$;
    * Return $x_{k_i}^{(j)}, P_{id_i}^{(j)} := (I_{id_i}^{\mathsf{left},(j)}, I_{id_i}^{\mathsf{right},(j)}, \mathcal{J}_{id_i}^{\mathsf{left},(j)}, \mathcal{J}_{id_i}^{\mathsf{right},(j)}, t_{id_i}^{(j)}, v_{id_i}^{(j)})$;
  - Upon receiving $(\text{FETCH}, \mathsf{sid}, \mathbf{x}^{(j)}, i^{(j)})$ from the environment $\mathcal{Z}$, player $S_j, j \in \{3,4\}$ does:
    * Set $w_1^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}(\mathsf{sid},1)$, $w_2^{(j)} \leftarrow \mathsf{PRF}_{\eta_{j-2}}(\mathsf{sid},2)$;
    * Set $\delta_{k,1}^{(j)} := i^{(j)} - \varphi_1^{(j-2)} - w_1^{(j)} \pmod n$, $\delta_{k,2}^{(j)} := i^{(j)} - w_2^{(j)} \pmod n$;
    * Set $\delta_{\mathsf{id},1}^{(j)} := \mathsf{id}_i^{(j)} - \psi_{i,1}^{(j)} - w_1^{(j)} \pmod{\widetilde{m}}$, $\delta_{\mathsf{id},2}^{(j)} := \mathsf{id}_i^{(j)} - w_2^{(j)} \pmod{\widetilde{m}}$;
    * Send $(\delta_{k,1}^{(j)}, \delta_{\mathsf{id},1}^{(j)})$ to S_1 and S_2, $(\delta_{k,2}^{(j)}, \delta_{\mathsf{id},2}^{(j)})$ to S_{7-j};
  - Upon receiving $(\delta_{k,2}^{(1)}, \delta_{\mathsf{id},2}^{(1)})$ from S_1, $(\delta_{k,2}^{(2)}, \delta_{\mathsf{id},2}^{(2)})$ from S_2, and $(\delta_{k,2}^{(7-j)}, \delta_{\mathsf{id},2}^{(7-j)})$ from S_{7-j}, player $S_j, j \in \{3,4\}$ does:
    * Set $\delta_{k,2} := \delta_{k,2}^{(1)} + \delta_{k,2}^{(2)} + \delta_{k,2}^{(3)} + \delta_{k,2}^{(4)} \pmod n$, $\widetilde{x}_q^{(j)} := x_{q-\delta_{k,2} \pmod n}^{(j)}$, for $q \in \mathbb{Z}_n$;
    * Set $\delta_{\mathsf{id},2} := \delta_{\mathsf{id},2}^{(1)} + \delta_{\mathsf{id},2}^{(2)} + \delta_{\mathsf{id},2}^{(3)} + \delta_{\mathsf{id},2}^{(4)} \pmod{\widetilde{m}}$, $\widetilde{P}_q^{(j)} := P_{q-\delta_{\mathsf{id},2} \pmod{\widetilde{m}}}^{(j)}$, for $q \in \mathbb{Z}_{\widetilde{m}}$;
    * Set $\zeta_{x,j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},1)$, $\zeta_{x,4} \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},1)$, $\zeta_{t,j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},2)$, $\zeta_{t,4} \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},2)$, $\zeta_{v,j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},3)$, $\zeta_{v,4} \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{2^\ell}}(\mathsf{sid},3)$, $\zeta_{I_1,j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},1)$, $\zeta_{I_1,4} \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},1)$, $\zeta_{I_2,j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},2)$, $\zeta_{I_2,4} \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_{\widetilde{m}}}(\mathsf{sid},2)$, $\zeta_{\mathcal{J}_1,j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_n}(\mathsf{sid},1)$, $\zeta_{\mathcal{J}_1,4} \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_n}(\mathsf{sid},1)$, $\zeta_{\mathcal{J}_2,j-2} \leftarrow \mathsf{PRF}_{\eta_{j-2}}^{\mathbb{Z}_n}(\mathsf{sid},2)$, $\zeta_{\mathcal{J}_2,4} \leftarrow \mathsf{PRF}_{\eta_4}^{\mathbb{Z}_n}(\mathsf{sid},2)$;
    * Set $(\beta_{q,\varphi_2}^{(j)})_{q\in\mathbb{Z}_n} \leftarrow \mathsf{DPF.EvalAll}(j-2, \mathcal{K}_{i,\varphi_2}^{(j-2)})$ and $(\beta_{q,\phi_2}^{(j)})_{q\in\mathbb{Z}_{\widetilde{m}}} \leftarrow \mathsf{DPF.EvalAll}(j, \mathcal{K}_{i,\phi_2}^{(j)})$;
    * Set $x_{k_i}^{(j)} := \sum_{q=0}^{n-1}(\widetilde{x}_q^{(j)} \cdot \beta_{k,\varphi_1}^{(j)}) - \zeta_{x,j-2} + (-1)^j \cdot \zeta_{x,4}$.
    * Set $I_{\mathsf{id}_i}^{left,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),left} \cdot \beta_{q,\phi_2}^{(j)}) - \zeta_{I_1,j-2} + (-1)^j \cdot \zeta_{I_1,4}$, $I_{\mathsf{id}_i}^{right,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),right} \cdot \beta_{q,\phi_2}^{(j)}) - \zeta_{I_2,j-2} + (-1)^j \cdot \zeta_{I_2,4}$;
    * Set $\mathcal{J}_{\mathsf{id}_i}^{left,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),left} \cdot \beta_{q,\phi_2}^{(j)}) - \zeta_{\mathcal{J}_1,j-2} + (-1)^j \cdot \zeta_{\mathcal{J}_1,4}$, $\mathcal{J}_{\mathsf{id}_i}^{right,(j)} := \sum_{q=0}^{n-1}(\widetilde{I}_q^{(j),right} \cdot \beta_{q,\phi_2}^{(j)}) - \zeta_{\mathcal{J}_2,j-2} + (-1)^j \cdot \zeta_{\mathcal{J}_2,4}$;
    * Set $t_{\mathsf{id}_i}^{(j)} := \sum_{q=0}^{n-1}(\widetilde{t}_q^{(j)} \cdot \beta_{q,\phi_2}^{(j)}) - \zeta_{t,j-2} + (-1)^j \cdot \zeta_{t,4}$, $v_{\mathsf{id}_i}^{(j)} := \sum_{q=0}^{n-1}(\widetilde{v}_q^{(j)} \cdot \beta_{q,\phi_2}^{(j)}) - \zeta_{v,j-2} + (-1)^j \cdot \zeta_{v,4}$;
    * Return $x_{k_i}^{(j)}, P_{id_i}^{(j)} := (I_{id_i}^{\mathsf{left},(j)}, I_{id_i}^{\mathsf{right},(j)}, \mathcal{J}_{id_i}^{\mathsf{left},(j)}, \mathcal{J}_{id_i}^{\mathsf{right},(j)}, t_{id_i}^{(j)}, v_{id_i}^{(j)})$.

Figure 18: Selection protocol.